Llenguatges de consulta en l'àmbit de la Web Semàntica

Cas d'estudi: SPARQL

MEMÒRIA
TREBALL FI DE CARRERA

Autor: Aleix Jordana Canedo
Consultor: Òscar Celma Herrada
Enginyeria Tècnica Informàtica Sistemes
Treball Fi de Carrera
11 de gener de 2007

ÍNDEX

1. INTRODUCCIÓ

1.1. MOTIVACIÓ

Actualment, les bases de dades estan evolucionant davant l'aparició de nous reptes en la gestió de la informació, un d'ells relacionat amb l'estàndard *XML* (*eXtensible Markup Language*) i la ***Web Semàntica***, una web amb més dotació de significat que l'actual com es veurà en el capítol 2 d'aquest document.

La proliferació de la *web* comporta la necessitat de tractar dades semiestructurades i avui en dia el llenguatge *XML* es fa servir per intercanviar informació entre diferents sistemes d'informació donada la flexibilitat i independència de fonts de dades i de formats fixes que proporciona.

L'objectiu que es persegueix és que els ordinadors puguin entendre el significat (semàntica) de la informació que hi ha a la *web* i sigui possible integrar diverses fonts d'informació, malgrat la procedència i formats de cadascuna d'elles: s'està dissenyant una *Web Semàntica*. Aquesta nova àrea de recerca està clarament lligada amb el món de les bases de dades, les quals han d'oferir (i de fet algunes ja ofereixen) un nou ventall de possibilitats per poder tractar, emmagatzemar i gestionar aquest nou tipus d'informació semiestructurada, alhora que permetin la comunicació entre sistemes de manera transparent a l'usuari.

Per totes aquestes raons, recentment han aparegut nous llenguatges de consulta, com ara ***SPARQL*** que és una proposta d'estàndard del W3C, els quals permeten gestionar la informació dins l'àmbit de les bases de dades *XML* i la *Web Semàntica*. En el capítol 3 es definirà aquest llenguatge i se'n veuran exemples.

En l'àmbit de la *Web Semàntica*, la informació es codifica seguint la notació ***RDF*** (*Resource Description Framework*). El llenguatge *RDF* proveeix interoperabilitat entre les aplicacions que intercanvien informació a través de la web. En l'apartat 2.3 es podrà veure aquest llenguatge i diversos exemples i representacions.

En concret, s'estudiarà el llenguatge de consultes *SPARQL*, així com els diferents **SGBD** que el suporten. El llenguatge *SPARQL* permet realitzar consultes sobre documents *RDF*.

El projecte es basa en estudiar i avaluar diferents sistemes gestors de bases de dades (SGBD) per desar informació dins del context de la *Web Semàntica.*, tal com es veurà en el capítol 4. La *Web Semàntica* permet dotar de significat al contingut textual de la *web*, permetent que sigui interpretable per una màquina.

En el capítol 5 es presentarà un cas d'estudi concret, on es proposa que el conjunt de dades sobre els quals es realitzin les consultes *SPARQL* siguin dades relacionades amb el projecte *Friend of a Friend* (FOAF). L'objectiu de la iniciativa FOAF és descriure la informació referent a les persones, destinada a ésser processada a l'entorn web.

1.2. OBJECTIUS

El objectius del present treball són els següents:

a. Estudiar els conceptes bàsics de la *Web Semàntica*.

b. Conèixer l'estructura i representació del llenguatge *RDF* de representació de la informació a la web.

c. Conèixer l'estructura i organització dels SGBD que treballen amb informació basada en RDF.

d. Avaluar l'adequació d'utilització dels SGBD per desar i recuperar descripcions en RDF.

e. Anàlisi comparatiu exhaustiu dels diferents (almenys dos) SGBD estudiats a l'objectiu anterior.

f. Estudi del llenguatge de consulta SPARQL.

g. Realització de casos pràctics d'aquestes tecnologies, implementant una aplicació senzilla que permeti fer cerques sobre documents RDF, amb dades relacionades amb la iniciativa FOAF.

2. WEB SEMÀNTICA

2.1. INTRODUCCIÓ

Des de l' aparició l'any 1989 de la *World Wide Web*, la xarxa ha transformat la manera de comunicar-nos provocant una revolució en la nostra societat, comparable a la d'altres grans mitjans com la radio, la televisió o el telèfon. La web és avui un mitjà extraordinàriament flexible i econòmic per a la comunicació, el comerç, els negocis, l'oci, l'entreteniment, l'accés a la informació i als serveis, la difusió...

Des de les primeres tecnologies bàsiques – el llenguatge HTML (*HyperText Markup Language*) i el protocol HTTP (*HyperText Transfer Protocol*) – fins als nostres dies, han sorgit tecnologies com CGI, Java, JavaScript, ASP, JSP, PHP, Flash, J2EEE o XML entre d'altres, que permeten una web més àmplia, potent, flexible i més fàcil de mantenir. La generació dinàmica de pàgines, l'acoblament amb bases de dades, la major interactivitat amb l'usuari, la concepció de la web com plataforma universal per al desplegament d'aplicacions o l'adaptació amb l'usuari, són algunes de les tendències evolutives més marcades dels últims anys.

Avui en dia gairebé tot està representat d'alguna manera a la web i amb l'ajuda d'un cercador podem trobar informació sobre gairebé qualsevol cosa que necessitem. La web està a prop de convertir-se en una enciclopèdia universal del coneixement humà. I també ens permet realitzar diferents activitats de la nostra vida quotidiana.

Però l'enorme volum que ha adquirit la web – que n'és una de les claus del seu èxit – fa que algunes tasques siguin inabastables per a una persona o que li requereixin un temps excessiu. El contingut de les pàgines generades automàticament des de bases de dades fa que la web es presenti sense la informació estructurada original que es troba en aquestes bases de dades.

L'ús de la web actual consisteix en cercar i utilitzar informació, buscar i estar en contacte amb altres persones, mirar catàlegs on-line per comprar productes, omplir formularis, etc. Aquestes activitats no són ben suportades per les eines de software actuals com per exemple els buscadors, tant imprescindibles avui en dia. Quan fem una cerca en buscadors com AltaVista, Yahoo o Google ens podem trobar amb les següents problemàtiques:

- **Baixa precisió dels resultats**: es troben gran multitud de pàgines irrellevants de les quals només unes poques mostren el que realment estem buscant.

- **Pocs o cap resultat**: pot succeir que no es tingui resposta a la nostra petició o que les pàgines amb informació rellevant no se'ns mostrin, encara que la manca de resultats és el menys freqüent en els resultats d'una cerca dels buscadors actuals.

- **Alta sensibilitat del vocabulari en els resultats:** sovint la paraula amb la que hem realitzat la cerca no ens mostra els resultats que desitgem. En aquests casos els documents rellevants utilitzen una terminologia diferent al de la consulta original. Això no és el que desitjaríem ja que consultes amb semàntiques similars voldríem que retornessin semblants.

- **Els resultats es troben a pàgines web simples**: Si la informació que desitgem trobar s'estén per diferents pàgines hem de realitzar diverses consultes per tal de trobar les pàgines que la poden contenir, i després extreure "manualment" les parts de la informació de cadascuna d'elles.

Malgrat les millores en la tecnologia dels buscadors, el gran volum d'informació a la xarxa creix més ràpidament que el progrés tecnològic dels buscadors i són diversos els aspectes susceptibles a millorar.

I si malgrat tot el resultat de la cerca és positiu, serà l'usuari qui haurà de navegar entre els resultats per extreure la informació amb el cost temporal que li comporta la manca de suport per extreure la informació.

Per exemple, si volem trobar "un article *sobre* Gabriel Garcia Márquez" trobarem multitud d'articles *de* Gabriel Garcia Márquez però hauríem d'accedir a cada pàgina per trobar-ne algun que tracti sobre aquest autor.

El principal obstacle per proveir un millor suport als usuaris de la web és la falta de capacitat de les representacions en què es basa la web actual per expressar significats. Els continguts i serveis de la web es presenten en format HTML comprensibles per a les persones però no per a les màquines.

L'exemple següent mostra aquesta situació amb una versió simplificada d'una pàgina d'informació metereológica. Mentre que la presentació de les dades a través del navegador és fàcilment comprensible per una persona, es molt difícil per a una ordinador "entendre" la temperatura, l'estat del cel i la resta de la semàntica en el codi de la pàgina.


```
<html><head><title>El tiempo en
Barcelona - la previsión del tiempo
hoy</title></head>
<BODY><center><table border=0
cellpadding="0" cellspacing="1"
width="750"> <tr><td valign=top
width="99%"><table border=0
cellpadding="4" cellspacing="0"
width="100%"><tr><td valign=top
bgcolor="#9999cc"><font face=Arial>
<big><b>Yahoo! Tiempo -
Barcelona</b> </big></font><br>
<br><b><font face=Arial size ="-
1">C<big>&#176;</big> o <a
href="index_f.html">F</a><big>&#176
;</big></font></b></td><td
bgcolor="#9999cc"
align=right><iframe_src="http://es.adserver.yahoo.com/a?f=150601698&p=es&l=MH&
c=h" width=60 height=10 marginwidth=0 marginheight=0 hspace=0 vspace=0
frameborder=0 scrolling=no bordercolor=000000></iframe>
</noscript></td></tr><tr><td valign="top"><font size=-1 face=Arial><b><a
href="/">Tiempo </a> &gt; <a href="/Europa/Espana/index.html">España</a> &gt;
Barcelona</b> </font></td><td align=right></td></tr></table></td><script
language="JavaScript" type="text/javascript"
src="http://es.adserver.yahoo.com/a?f=150601698&p=es&l=NE&c=r"></script><noscr
ipt><iframe src="http://es.adserver.yahoo.com/a?f=150601698&p=es&l=NE&c=h"
width=60 height=10 marginwidth=0 marginheight=0 hspace=0 vspace=0
frameborder=0 scrolling=no bordercolor=000000></iframe></noscript></tr>
</table></table></BODY></HTML>
```

2.2. LA WEB SEMÀNTICA

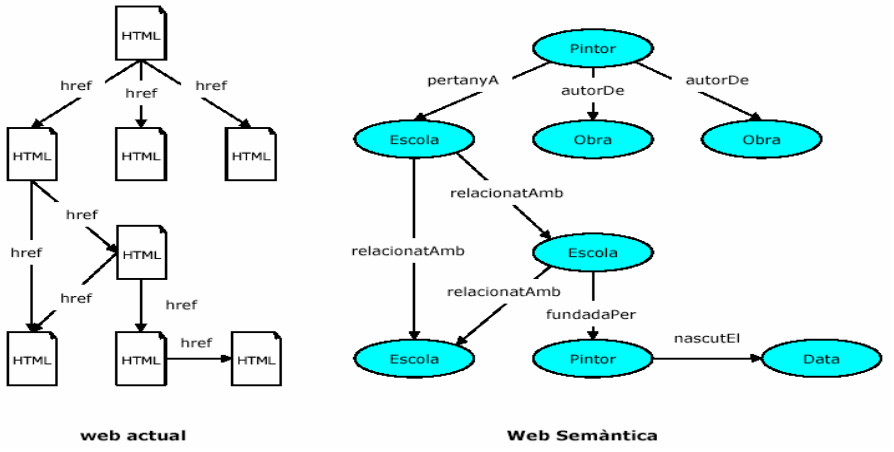
Entre les últimes tendències que poden repercutir en el futur de la web a mig termini, a finals dels anys 90 apareix la visió del que s'ha denominat la Web Semàntica[1]. Es tracta d'un corrent promogut per Sir Tim Berners-Lee (inventor de la WWW i president del *World Wide Web Consortium*[2]) que pretén aconseguir que les màquines puguin arribar a entendre i utilitzar el contingut de la web introduint les descripcions explícites del significat, l'estructura interna i l'estructura global dels continguts i serveis disponibles a la xarxa.

La Web Semàntica és una extensió de la web actual, afegint un major significat als recursos disponibles. Per tant, qualsevol usuari pot trobar resposta a les seves consultes de manera més ràpida i senzilla gràcies a una informació més ben definida. Al proveir la web de més significat, i per tant, de semàntica, es poden obtenir solucions a problemes habituals com la cerca d'informació gràcies a la utilització d'una infrastructura comú que permet compartir, processar i transferir informació d'una forma més senzilla. Aquesta nova web, estaria poblada per agents capaços de navegar i realitzar operacions per nosaltres per tal d'estalviar-nos feina i optimitzar resultats.

Les tecnologies de la Web Semàntica busquen desenvolupar una web més cohesionada a on sigui encara més fàcil localitzar, compartir i integrar informació i serveis per treure un partit encara més gran dels recursos disponibles a la web.

La Web Semàntica proposa superar les limitacions de la web actual mitjançant la introducció de descripcions explícites del significat, l'estructura interna i l'estructura global dels continguts i dels serveis disponibles a la xarxa. Davant de la semàntica implícita, el creixement caòtic de recursos i l'absència d'organització de la web actual, la Web Semàntica aposta per classificar, dotar d'estructura i anotar els recursos amb una semàntica explícita processable per les màquines.

Mentre que la web actual es pot representar com un graf format per nodes del mateix tipus i arcs (hiperenllaços) també indiferenciats, a la Web Semàntica cada node (recurs) és d'un tipus determinat i els arcs representen relacions explícitament diferenciades. En la figura següent es pot veure com es representaria la informació sobre pintura en la que es defineixen relacions entre pintors, obres i escoles de pintura en la web actual i en la Web Semàntica.



web actual **Web Semàntica**

[1] http://www.w3.org/2001/sw/
[2] http://www.w3.org/

La Web Semàntica conserva els principis que han fet un èxit de la web actual com són: els principis de descentralització, compartició, compatibilitat o màxima facilitat d'accés i contribució. En aquest context un problema clau és aconseguir un enteniment entre les parts que han d'intervenir en la construcció i explotació de la web: usuaris, programadors i programes. Per a aconseguir-ho la Web Semàntica rescata la idea de ontologia del camp de la Intel·ligència Artificial com vehicle per assolir aquest objectiu.

Ontologia és un terme originari de la branca metafísica de la filosofia encarregada d'estudiar la naturalesa de l'existència i com descriure-la. En el camp de la informàtica una ontologia es defineix com una especificació formal i explícita d'una conceptualització i descriu formalment un domini.

La idea és que la Web Semàntica estigui formada per una xarxa de nodes tipificats i interconnectats mitjançant classes i relacions definides per una ontologia compartida pels seus diferents autors. Per exemple, un cop establerta una ontologia sobre quadres i pintures, un museu virtual podria organitzar els seus continguts definint instàncies de pintors i quadres, interrelacionant-les i publicant-les a la Web Semàntica. L'adopció d'ontologies comuns és la clau per que tots els participants de la Web Semàntica puguin treballar de manera autònoma. D'aquesta manera diversos museus podrien treballar units per crear un gran meta-museu que integri els continguts de tots ells.

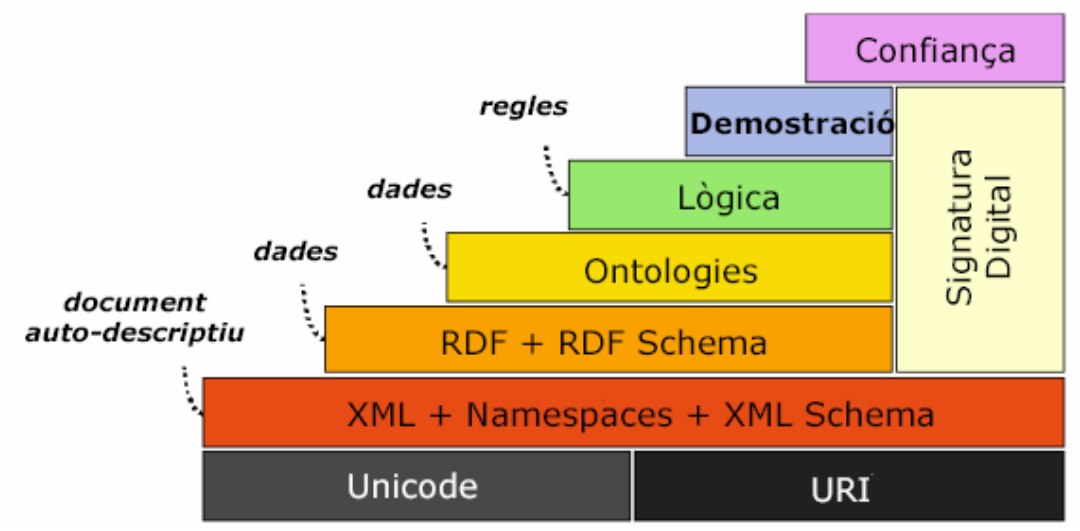
La web no només ofereix accés a continguts sinó que també proporciona interaccions i serveis. Els serveis són una línia important de la Web Semàntica que no només proposa descriure informació sinó també definir ontologies de funcionalitat i procediments per descriure serveis web: les seves entrades i sortides, les condicions necessàries per que es puguin executar, els efectes que produeixen, o les passes a seguir quan es tracta d'un servei compost. Aquestes descripcions processables per màquines permetran automatitzar el descobriment, la composició i l'execució de serveis, així com la comunicació entre uns i altres.

La definició més acceptada de Serveis Web (existeixen nombroses definicions, el que demostra la gran complexitat d'aquests serveis) és que són un conjunt d'aplicacions o tecnologies amb capacitat per interoperar a través de la web. Aquestes tecnologies intercanvien dades entre elles amb el propòsit d'oferir serveis. Els proveïdors ofereixen els seus serveis com procediments remots i els usuaris els sol·liciten cridant-ne els procediments a través de la web.

Actualment el W3C està treballant per dissenyar l'arquitectura, definir-la i crear un nucli de tecnologies que facin possibles els Serveis Web Semàntics. El *Web Services Description Working Group*[3] del W3C va publicar el passat juny de 2006 el llenguatge WSDL (*Web Services Description Language*) com a proposta d'estàndard per als Serveis Semàntics.

[3] *http://www.w3.org/2002/ws/desc/*

La Web Semàntica implementa el principis definits pel W3C en forma de capes de tecnologies estàndards i web tal com es pot veure en el següent esquema:



- Les capes **Unicode** i **URI** asseguren que es faran servir conjunts de caràcters internacionals i proporciona els mitjans per identificar els objectes dins la Web Semàntica.

- La capa **XML**, que inclou els espais de noms i les definicions d'esquemes, assegura la integració de les definicions que es facin a la Web Semàntica amb d'altres estàndards basats en el llenguatge XML.

- La capa **RDF** i **RDF Schema** permet la creació d'axiomes sobre els objectes i la definició de vocabularis que es poden identificar mitjançant una URI. En aquesta capa es poden assignar tipus als recursos i als enllaços.

- La capa **Ontologies** suporta l'evolució dels vocabularis definits a la capa inferior, ja que permet la definició de relacions entre diferents conceptes.

- La capa de **Signatura Digital** permet la detecció de modificacions en els documents.

- Les capes **Lògica**, **Demostració** i **Confiança** estan en fase d'estudi. Avui dia només existeixen aplicacions simples que serveixen com a demostradores d'aquestes tecnologies. L'objectiu d'aquestes capes és el següent: la capa Lògica permet la definició de regles, mentre que la capa Demostració .

Aquest treball es centrarà en les capes **RDF** i **RDF Schema** (apartats 2.3 i 2.4) i se'n veuran exemples a partir d'una ontologia concreta. També es veurà la capa **Ontologies** en l'apartat 2.5, en la que es farà una descripció del llenguatge OWL per definir ontologies.

2.3. EL LLENGUATGE RDF

El llenguatge RDF[4] (*Resource Definition Framework*) és una proposta del W3C per al nou model d'estructuració de la informació necessari per crear la Web Semàntica. L'RDF defineix un model de dades que suporta de manera fàcil i ràpida la integració de diferents fonts d'informació utilitzant conceptes semàntics.

Per a definir els recursos de RDF utilitza les URIs. Una URI (*Uniform Resource Identifier*) és un mecanisme simple i extensible per identificar un recurs i consisteix en una cadena de caràcters que permet identificar un recurs independentment del context en què es trobi. Les URLs (*Uniform Resource Locator*) són un subconjunt de les URIs que, a més d'identificar un recurs, proporcionen els mitjans per localitzar-lo a través d'internet.

RDF és un model per representar propietats i els seus valors. Les propietats RDF poden recordar a atributs de recursos, i en aquest sentit es corresponen amb els tradicionals parells d'atribut-valor. Les propietats RDF també representen les relacions entre recursos i per tant *grosso-modo*, un model RDF es pot assimilar a un diagrama d'entitat-relació.

RDF es basa la idea de convertir les declaracions dels recursos de la web en expressions (tripletes) del tipus **subjecte–predicat–objecte** tal com es pot veure en els tres tipus d'elements que composen el model:

- **Recurs:** qualsevol cosa que pot ser representada per una URI. Pot ser una pàgina web, una part d'una pàgina o un objecte no directament accessible des de la web, com per exemple un llibre, un lloc, una persona, un hotel, etc.

- **Propietat:** un recurs que té un nom i pot utilitzar-se com una propietat, com per exemple un autor o un títol. Una propietat és un aspecte específic, característica, atribut o relació utilitzat per descriure un recurs. Cada propietat té un significat específic, defineix els seus valors permesos, els tipus de recursos que pot descriure i les seves relacions amb altres propietats.

- **Sentència:** és la combinació d'un recurs, una propietat i un valor (sentència RDF). Aquestes tres parts individuals d'una sentència s'anomenen **subjecte**, **predicat** i **objecte**. El valor pot ser un literal o bé un altre recurs (una URI). Un exemple de sentència seria "L'autor de http://www.textuality.com/RDF.why.html és Tim Bray".

Per tant una sentència RDF conté: un **subjecte** (el recurs que es vol descriure), un **predicat** (la relació o propietat que es desitja definir per al subjecte) i un **objecte** (el valor de la propietat).

RDF està dissenyat per tenir les següents característiques:

- **Independència:** donat que una propietat és un recurs, tota organització independent o fins i tot cada persona pot inventar-les.

[4] http://www.w3.org/RDF/

- **Intercanvi:** com que les sentències RDF es poden descriure amb XML[5] poden ser fàcilment utilitzades per intercanviar informació.

- **Escalabilitat:** les sentències RDF son simples, registres de tres camps (recurs, propietat, valor) per tant són fàcils d'utilitzar per buscar objectes en volums grans.

- **Les propietats són recursos:** les propietats poden tenir a l'hora les seves propietats i poden ser trobades i manipulades com qualsevol altre recurs.

- **Els valors poden ser recursos:** les sentències també tenen propietats.

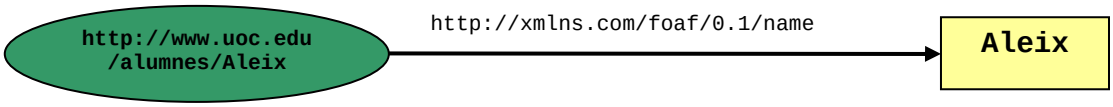
El model de dades de l'RDF recorda vagament al paradigma de l'orientació a objecte. Està format per **entitats**, representades per un identificador únic URI, i per relacions binàries, anomenades **sentències**, que relacionen entitats entre sí. En la terminologia del disseny orientat a objectes els recursos es corresponen a objectes, les propietats es corresponen a objectes específics i variables d'una categoria.

El model de dades diferencia els recursos, que són objectes que tenen associat una URI, dels literals, que són simples cadenes de caràcters. El subjecte i la propietat d'una sentència són sempre recursos, mentre que l'objecte pot ser també un recurs, o bé, un literal.

Una manera de representar recursos RDF pot ser amb una taula identificant la tripleta subjecte–predicat–objecte:

SUBJECTE	PREDICAT	OBJECTE
`http://www.uoc.edu/alumnes/Aleix`	`http://xmlns.com/foaf/0.1/name`	`Aleix`

RDF gràficament es pot representar com un graf etiquetat dirigit[6] en el que es connecten els dos nodes corresponents al subjecte i l'objecte mitjançant un arc dirigit que representa el predicat. La sentència anterior es representaria de la següent



manera:

El recurs (entitat) identificat per l'URI `http://www.uoc.es/alumnes/Aleix` està relacionada amb el literal `Aleix` mitjançant una propietat identificada per l'URI `http://xmlns.com/foaf/0.1/name`. En llenguatge natural es diria que el recurs `Aleix` té per nom `Aleix`.

La sintaxi per declarar espais de noms en XML com abreviatura de URIs pot ser aplicada en RDF per abreujar les URIs en les sentències. En l'exemple anterior es pot substituir `http://xmlns.com/foaf/0.1/name` per un prefix d'espai de noms `foaf`. Això dóna com a resultat sentències més simples (`foaf:nom`). Una de les abreviatures que s'acostuma a fer servir és el prefix `rdf` per referir-se al vocabulari

[5] http://www.w3.org/TR/rdf-syntax-grammar/ conté l'especificació de la sintaxi RDF/XML
[6] *Les especificacions relatives a RDF del W3C indiquen que els nodes dels recursos cal representar-los com un oval verd, el objectes literals com rectangles grocs, i els arcs com connexions direccionades que van del subjecte a l'objecte de cadascuna de les sentències.*

definit en l'especificació de la sintaxi i el model del llenguatge RDF. Aquest prefix es correspon a l'espai de noms `http://www.w3.org/1999/02/22-rdf-syntax-ns#`.

A RDF és possible assignar cada recurs a un tipus concret. Per exemple, en els exemples anteriors, podria ser útil indicar que el recurs `Aleix` és de tipus `Alumne`. De la mateixa manera, el tipus `Alumne` també podria ser un subtipus de `Persona`.

A continuació exemplificarem el que hem vist fins al moment usant com a exemple la iniciativa FOAF[7] (*Friend Of A Friend*) que és una especificació d'una semàntica en RDF (ontologia) per descriure les persones.

Una de les classes bàsiques definides dins de l'ontologia FOAF és `Person` i sota les instàncies d'aquesta classe s'associen propietats mitjançant predicats com ara el nom (`name`), el cognom (`firstName`), l'adreça de correu electrònic (`mail`), els enllaços amb altres persones (`knows`) o fins i tot el seu gènere (`gender`), entre d'altres.

Un dels primers problemes a resoldre a l'hora de definir FOAF va ser com assignar URIs a les persones de manera que pugessin ser identificades de manera unívoca. Utilitzar directament el seus nom i cognoms per composar una URI no semblava una bona opció quan es volia crear una aplicació d'àmbit mundial (existeix una probabilitat en cap cas negligible que dues persones es diguin de la mateixa manera). Després d'analitzar el problema es va arribar a la conclusió que la URL associada a la seva adreça de correu electrònic era més apropiada, doncs els serveis de correu electrònic acostumen a ser de caràcter personal i intransferible. No obstant es plantejava un problema ja que si el conveni acceptat era que l'adreça de correu fos la URI de la persona, es desproveïa de mecanismes per diferenciar l'adreça de correu en sí mateixa com una entitat independent.

Es a dir, suposem el cas en que un virus informàtic que es transmet per correu electrònic es propaga dins d'una intranet corporativa. Al existir un registre RDF de les incidències succeïdes en el que constaria una llista de les adreces a les que ha afectat , trobaríem sentències dels següent estil:

SUBJECTE	PREDICAT	OBJECTE
El virus ZX27	ha infectat a	ajordana@uoc.edu

En el cas que la URI de la persona sigui la seva adreça la sentència equival a dir que aquesta persona ha estat afectada pel virus quan en realitat es vol dir que el virus s'ha propagat a través del seu correu i per tant el que pot estar afectat és el seu software enlloc del seu propi organisme.

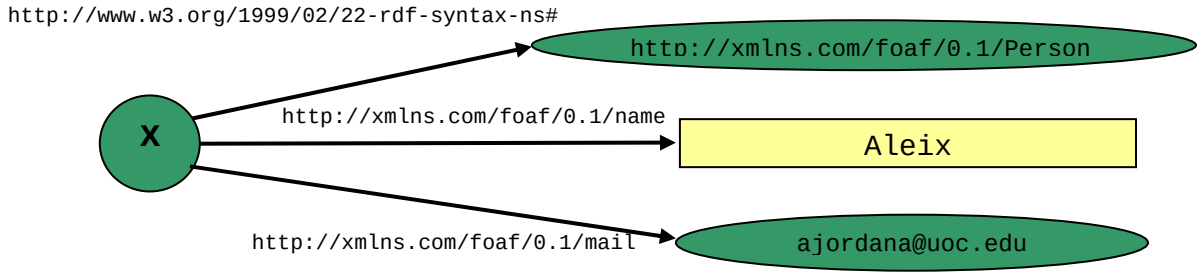
FOAF aporta una solució per aquest problema de manera que en cap moment s'assigna una URI a la persona sinó que defineix el que es coneix com a node buit del graf. La proposta consisteix en establir jocs de sentències del següent tipus:

SUBJECTE	PREDICAT	OBJECTE
X	pertany a la classe	persona
X	es diu	Aleix
X	té per adreça de correu	ajordana@uoc.edu

[7] http://www.foaf-project.org/

On X representa a un node buit.

La representació gràfica de les sentències anteriors seria la següent:



Els sistemes RDF tenen un URI assignat (que és on es troba la definició de l'ontologia), i en el cas de FOAF es tracta de `http://xmlns.com/foaf/0.1/`. L'URI de cadascun dels subjectes que descriu s'obté afegint una paraula clau al final del recurs general FOAF. Com per exemple `http://xmlns.com/foaf/0.1/person` que defineix l'URI del recurs associat a la classe `Person` de FOAF.

Per expressar el predicat que en llenguatge natural expressaríem al dir que una persona coneix a una altra té una representació dins de l'esquema FOAF en el que l'URI assignat és `http://xmlns.com/foaf/0.1/`**knows**.

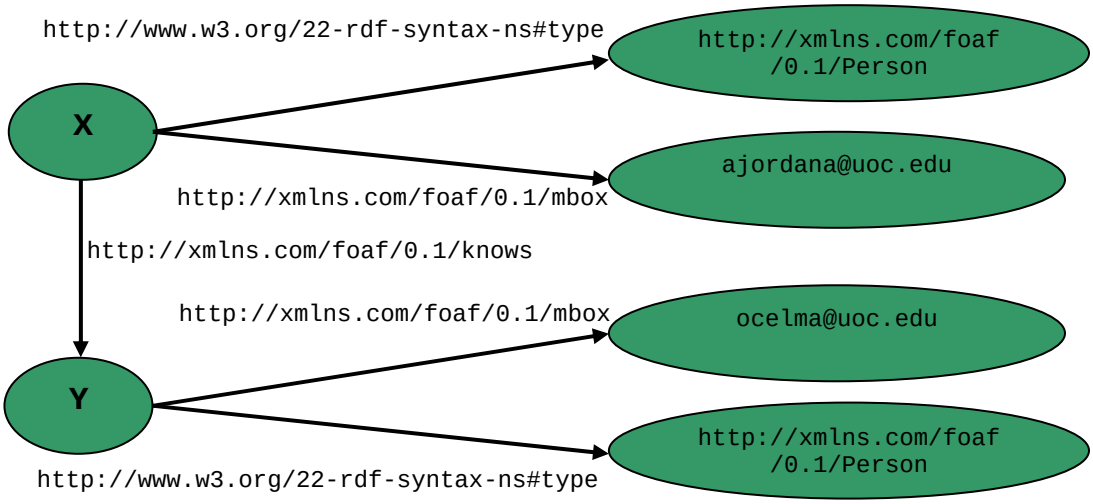
Per tant la relació de dues persones es representaria de la següent manera:

SUBJECTE	PREDICAT	OBJECTE
X	`http://www.w3.org/22-rdf-syntax-ns#type`	`http://xmlns.com/foaf/0.1/Person`
X	`http://xmlns.com/foaf/0.1/mbox`	`ajordana@uoc.edu`
Y	`http://www.w3.org/22-rdf-syntax-ns#type`	`http://xmlns.com/foaf/0.1/Person`
Y	`http://xmlns.com/foaf/0.1/mbox`	`ocelma@uoc.edu`
X	`http://xmlns.com/foaf/0.1/knows`	Y

Si analitzem la composició anterior, tenint en compte que `http://www.w3.org/22-rdf-syntax-ns#type` es tracta de la URI associada al concepte de pertinença a una classe establert en el sistema RDF, podem veure que el llenguatge natural equivalent és el següent:

i) existeixen dos recursos que pertanyen a la classe persona que són X i Y.
ii) un d'ells té l' adreça de correu electrònic `ajordana@uoc.edu`
iii) l'altre té l' adreça de correu electrònic `ocelma@uoc.edu`
iv) les dues persones es coneixen

La representació gràfica de les sentències anteriors seria de la següent:



Un cop la informació està en format RDF és més fàcil que sigui processada. RDF és un format genèric que compta amb diverses representacions i existeixen diferents sintaxis que permeten realitzar aquesta tasca. La sintaxi que recomana el W3C és l'RDF/XML ja que permet aprofitar les nombroses eines existents dels documents XML.

La pregunta que ens podria sorgir en aquest punt és per que no es poden representar les dades directament amb XML. L'XML és un primer pas per fer una representació explícita de les dades i de l'estructura dels continguts de la web, separada de la seva representació en HTML. XML proporciona la sintaxi per a fer-ho possible però ofereix una capacitat limitada per expressar la semàntica. És per aquest motiu que s'ha creat la necessitat de definir RDF com a model de dades per representar la semàntica de la web. El model de dades d'XML consisteix en un arbre que no distingeix entre objectes i relacions, ni té nocions de jerarquies mentre que RDF com ja hem vist és més potent ja que és representat per un graf dirigit.

A continuació veurem l'estructura bàsica d'un document RDF/XML. Tot document RDF/XML té l'element **`<rdf:RDF>`** que encapsula totes les sentències. Dins d'aquesta etiqueta es declaren els espais de noms que es faran servir mitjançant l'atribut **`xmlsns:`** seguit de l'etiqueta que identificarà aquest espai de noms. Per defecte sempre es declara l'espai de noms del vocabulari definit en l'especificació d'RDF amb `xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"`.

```
<?xml version="1.0"?>
<rdf:RDF
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:foaf=" http://xmlns.com/foaf/0.1/">
</rdf:RDF>
```

En l'exemple anterior s'han definit també l'espai de noms **`foaf`** associat a la URI `http://xmlns.com/foaf/0.1/name` que el conté.

Les sentències RDF es representen mitjançant l'etiqueta **`<rdf:Description>`**, i els tres components d'una sentència RDF es poden representar com s'indica a continuació:

- **Subjecte:** es representa amb l'atribut **`rdf:about`** dins de l'etiqueta `<rdf:Description>`. Per exemple, la sentència RDF on el subjecte està identificat per l'URI `http://www.uoc.edu/alumnes/Aleix` seria la següent:

```
<rdf:Description rdf:about="http://www.uoc.edu/alumnes/Aleix">
      ...
</rdf:Description>
```

- **Propietat:** es pot representar mitjançant un atribut de l'etiqueta `<rdf:Description>` o amb una nova etiqueta que és filla de l'anterior.

- **Objecte:** La representació de l'objecte de la sentència depèn de si s'ha representat la propietat com a atribut o com a etiqueta.

En el següent exemple es mostra la representació d'una propietat mitjançant una etiqueta i d'un objecte de tipus literal:

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:foaf=" http://xmlns.com/foaf/0.1/">
   <rdf:Description rdf:about="http://www.uoc.edu/alumnes/Aleix">
        <foaf:name>Aleix</foaf:name>
   </rdf:Description>
</rdf:RDF>
```

El mateix exemple anterior representat mitjançant un atribut dins l'etiqueta `<rdf:Description>` amb un objecte de tipus literal seria de la següent manera:

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:foaf=" http://xmlns.com/foaf/0.1/">
   <rdf:Description rdf:about="http://www.uoc.edu/alumnes/Aleix"
        foaf:name="Aleix"/>
</rdf:RDF>
```

Quan l'objecte en lloc de ser un literal és un recurs s'utilitza el literal `rdf:resource`. En el següent exemple es representa que l'Aleix, que té per adreça de correu `ajordana@uoc.edu`, coneix a l' Óscar:

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:foaf=" http://xmlns.com/foaf/0.1/">

   <rdf:Description rdf:about="http://www.uoc.edu/alumnes/Aleix"
        foaf:name="Aleix"
        foaf:mail="ajordana@uoc.edu">
        <foaf:knows rdf:resource="http://www.uoc.edu/tutors/Oscar">
   </rdf:Description>

   <rdf:Description rdf:about="http://www.uoc.edu/tutors/Oscar"
        foaf:name="Oscar"/>
</rdf:RDF>
```

En RDF/XML no importa l'ordre de definició de les sentències dins el document. Per tant la segona etiqueta `rdf:Description` podria estar abans que la primera sense que això afectés a la semàntica que s'està descrivint. L'exemple anterior es podria representar de forma compacta de la següent manera:

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:foaf=" http://xmlns.com/foaf/0.1/">

   <rdf:Description rdf:about="http://www.uoc.edu/alumnes/Aleix"
        foaf:name="Aleix"
        foaf:mail="ajordana@uoc.edu">
        <foaf:knows>
             <rdf:Description rdf:about="http://www.uoc.edu/tutors/Oscar"
                  foaf:name="Oscar"/>
        </foaf:knows>
   </rdf:Description>
</rdf:RDF>
```

En RDF/XML per representar el tipus de dades al qual ens estem referint es fa servir l'etiqueta `<rdf:type>`. En el següent exemple es mostra com indicar que el recurs `http://www.uoc.edu/alumnes/Aleix` és de tipus Alumne mitjançant aquesta etiqueta:

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   <rdf:Description rdf:about="http://www.uoc.edu/alumnes/Aleix">
        <rdf:type resource="http://www.uoc.edu/propietats/Alumne"/>
   </rdf:Description>
</rdf:RDF>
```

En les descripcions dels elements que defineixen un tipus es pot utilitzar el tipus especificat enlloc de `rdf:Description` per representar aquest sintaxi de forma més reduïda. En l'exemple anterior es faria de la següent manera:

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:propietats="http://www.uoc.edu/propietats/">
   <propietats:Alumne rdf:about=" http://www.uoc.edu/alumnes/Aleix"/>
</rdf:RDF>
```

En l'exemple anterior es substitueix l'etiqueta `rdf:Description` pel tipus que s'indicava a `rdf:type` com `propietats:Alumne`.

RDF defineix un model de dades simple per descriure propietats sobre els recursos i les relacions entre ells. No obstant, aquest model de dades no ofereix facilitats per definir tipus de dades específics ni classes de recursos. Tampoc no és possible definir quines propietats apliquen a una determinada classe de recursos. Per fer front a aquestes limitacions, el W3C ha definit l'RDF-Schema (RDFS).

2.4. EL LLENGUATGE RDFS

L'objectiu d'RDF-Schema[8] és especificar els mecanismes necessaris per definir recursos, classes, jerarquies de classes i restriccions en la utilització de classes i relacions. Per tant, ens permet definir un domini en particular.

RDF-Schema és similar al sistema de tipus dels llenguatges orientats a objectes en el sentit que, per exemple, permet que els recursos es puguin definir com a instàncies d'una o més classes. A més, permet que aquestes classes es puguin organitzar jeràrquicament. La principal diferència respecte els llenguatges orientats a objectes és que en lloc de definir les classes en termes de les propietats que tenen les seves instàncies, un esquema RDF defineix les propietats en termes de les classes de recursos a les quals s'aplica.

Les primitives bàsiques per descriure un esquema en RDF-Schema són les següents:

- **`rdfs:Class`** – permet definir una classe.
- **`rdfs:subClassOf`** – permet definir una subclasse.
- **`rdf:Property`** – permet definir una propietat.
- **`rdfs:subPropertyOf`** – permet definir una subpropietat. La combinació amb la primitiva anterior permet crear la definició de jerarquia de propietats de forma similar a la jerarquia de classes.
- **`rdfs:domain`** – permet definir un domini.
- **`rdfs:range`** – permet definir un rang.
- **`rdf:Type`** – permet declarar un recurs com una instància d'una classe determinada.

La definició de la classe `Alumne` es defineix de la següent manera:

[8] http://www.w3.org/TR/rdf-schema/

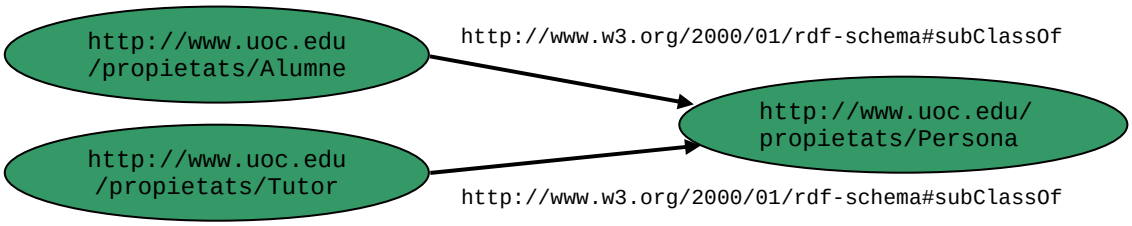
```
<rdfs:Class rdf:ID="Alumne">
   ...
</rdfs:Class>
```

La primitiva `<rdf:Description rdf:ID="Alumne">` defineix la classe Alumne on `rdf:ID="Alumne"` representa el nom de la classe que s'està creant.

I la combinació de les dues primitives permet poder crear relacions de jerarquia entre classes, com per exemple que un alumne és una subclasse de persona:

```
<rdfs:Class rdf:about="Alumne">
   <rdfs:subClassOf rdf:resource="Persona"/>
</rdfs:Class>
```

El següent esquema representa una jerarquia de classes on la superclasse està identificada per l'URI `http:www.uoc.edu/rdf/propietats/Persona` i representa al concepte Persona, que mitjançant la primitiva de RDF-Schema `http://www.w3.org/2000/01/rdf-schema#subClassOf` defineix les subclasses que representen els conceptes de Alumne i Tutor.



La representació en RDF/XML seria la següent:

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
   <rdf:Description rdf:ID="Persona">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
      <rdfs:subClassOf
            rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
   </rdf:Description>

   <rdf:Description rdf:ID="Alumne">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
      <rdfs:subClassOf rdf:resource="#Persona"/>
   </rdf:Description>

   <rdf:Description rdf:ID="Tutor">
      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
      <rdfs:subClassOf rdf:resource="#Persona"/>
   </rdf:Description>
</rdf:RDF>
```

El tipus del recurs s'especifica mitjançant l'etiqueta `rdf:type` acompanyada de la URI del recurs que s'està especificant. Com que es tracta de classes corresponen al recurs `http://www.w3.org/2000/01/rdf-schema#Class`. També s'indica la jerarquia mitjançant l'ús de l'etiqueta `rdfs:subClassOf`. En el cas de Tutor i Alumne, són subclasses de Persona i aquesta última és subclasse de la classe genèrica `http://www.w3.org/2000/01/rdf-schema#Resource`.

2.5. EL LLENGUATGE OWL

OWL (*Web Ontolohy Language*) és un llenguatge per a la definició d'ontologies estructurades basades en web que estén RDFS per a poder expressar relacions complexes entre diferents classes i ofereix una precisió més gran a l'hora d'establir restriccions a classes i propietats específiques. Va ser definit a partir del llenguatge DAM+OIL, que va sorgir a partir de la fusió dels llenguatges OIL (*Ontology Interface Language*) i DARML (*DARPA Agent Markup Language*) creats anteriorment.

Permet definir ontologies que poden ser utilitzades a través de diferents sistemes: els usuaris, les bases de dades i les aplicacions necessiten compartir informació específica d'una àrea de coneixement.

OWL té més funcionalitats per expressar el significat i la semàntica que els llenguatges anteriors:

- **XML** proporciona una sintaxis superficial per a documents estructurats, però no imposa cap restricció semàntica en el significats d'aquests documents.

- **XML Schema** és un llenguatge per restringir l'estructura dels documents XML i també amplia XML amb els `datatypes`.

- **RDF** és un model de dades per objectes (recursos) i les relacions entre aquests, proporciona una semàntica simple per aquest model de dades, i aquests models de dades es poden representar en sintaxis XML.

- **RDF Schema** és un vocabulari per descriure característiques i classes dels recursos RDF, amb una semàntica per la generalització/especialització d'aquestes característiques i classes.

OWL afegeix més vocabulari per descriure característiques i classes com són les relacions entre classes, cardinalitat, igualtat, enriqueix els tipus de propietats, les característiques de les propietats i les classes enumerades.

OWL conté tres subllenguatges d'expressió incremental dissenyats per l'ús específic dels seus usuaris. Serà cada usuari el que haurà de decidir quin dels tres llenguatges li cal utilitzar. Aquests llenguatges són:

- **OWL Lite**: dóna suport a aquells usuaris que principalment necessiten una classificació jeràrquica i restriccions simples. Per exemple, suporta restriccions de cardinalitats però només permet valors cardinals de 0 o 1.

- **OWL DL**: dóna suport a aquells usuaris que necessiten la màxima expressivitat mentre conserven completament la calculabilitat (totes les conclusions estan garantides per ser calculades) i resolució (tots els càlculs acabaran en un temps finit). OWL DL inclou tots els constructors del llenguatges OWL però només poden utilitzar-se sota certes restriccions (per exemple, metre una classe sigui subclasse d'alguna classe, aquesta classe no pot ser una instància d'una altra classe). S'anomena DL per a la seva correspondència amb *description logics*, un camp de la lògica investigat per la creació formal de OWL.

- **OWL Full**: dóna suport a aquells usuaris que requereixen la màxima expressivitat i llibertat sintàctica d'RDF sense garanties de càlcul. Per exemple, OWL Full una classe pot ser tractada simultàniament com una col·lecció d'individus i com un individu per dret propi. OWL Full permet a una ontologia augmentar el significat del vocabulari predefinit (RDF o OWL). És poc probable que cap software racional pugui suportar per complert el raonament per cada característica de OWL Full.

Cadascun d'aquest subllenguatges és una extensió del seu predecessor més simple. Uns ontologia pot ser definida com a legal i una conclusió com a válida si compleix les condicions següents (les condicions inverses no són vàlides):

- Cada ontologia OWL Lite legal és una ontologia OWL DL legal.
- Cada ontologia OWL DL legat és una ontologia OWL Full legal.
- Cada conclusió OWL Lite vàlida és una conclusió OWL DL vàlida.
- Cada conclusió OWL DL vàlida és una conclusió OWL Full vàlida.

OWL Full es pot veure com una extensió d'RDF, mentre que OWL Lite i OWL DL es poden veure com extensions restringides d'RDF. Cada document OWL (Lite, DL i Full) és document RDF i cada document RDF és un document OWL Full, però només alguns documents RDF seran documents legats OWL Lite o OWL DC. És per aquest motiu que s'ha de tenir especial cura a l'hora de migrar documents RDF a OWL.

A continuació es mostren els constructors de cadascun dels subllenguatges[9] de OWL:

- Llista de constructors de OWL Lite:

RDF Schema Features:
- Class (Thing, Nothing)
- rdfs:subClassOf
- rdf:Property
- rdfs:subPropertyOf
- rdfs:domain
- rdfs:range
- Individual

(In)Equality:
- equivalentClass
- equivalentProperty
- sameAs
- differentFrom
- AllDifferent
- distinctMembers

Property Characteristics:
- ObjectProperty
- DatatypeProperty
- inverseOf
- TransitiveProperty
- SymmetricProperty
- FunctionalProperty
- InverseFunctionalProperty

Property Restrictions:
- Restriction
- onProperty
- allValuesFrom
- someValuesFrom

Restricted Cardinality:
- minCardinality (only 0 or 1)
- maxCardinality (only 0 or 1)
- cardinality (only 0 or 1)

Header Information:
- Ontology
- imports

[9] *A* http://www.w3.org/TR/owl-features/ es *pot veure l'especificació detallada*

Class Intersection:
- IntersectionOf

Datatypes:
- xsd datatypes

Versioning:
- versionInfo
- priorVersion
- backwardCompatibleWith
- incompatibleWith
- DeprecatedClass
- DeprecatedProperty

Annotation Properties:
- rdfs:label
- rdfs:comment
- rdfs:seeAlso
- rdfs:isDefinedBy
- AnnotationProperty
- OntologyProperty

- Llista de constructors de OWL DL i OWL Full que amplien els de OWL Lite:

Class Axioms:
- oneOf, dataRange
- disjointWith
- equivalentClass (applied to class expressions)
- rdfs:subClassOf (applied to class expressions)

Boolean Combinations of Class Expressions:
- unionOf
- complementOf
- intersectionOf

Arbitrary Cardinality:
- minCardinality
- maxCardinality
- cardinality

Filler Information:
- hasValue

Fins al moment s'ha vist com es poden crear pàgines de la nova Web Semàntica expressant-ne el seu significat mitjançant RDF i utilitzant ontologies ja existents com per exemple FOAF. També s'ha vist com a través dels llenguatges RDF Schema i OWL es poden definir aquestes ontologies. Un cop idealitzada la xarxa de la Web Semàntica ens apareix la necessitat de veure com es poden realitzar consultes sobre aquesta nova web de tal manera que realment milloren les cerques sobre la web actual. Tal com veurem en el següent capítol el llenguatge de consultes SPARQL és un dels llenguatges per a les bases de dades que guarden dades RDF.

3. EL LLENGUATGE DE CONSULTES SPARQL

3.1. INTRODUCCIÓ

SPARQL[10] (*SPARQL Protocol and RDF Query Language*) és un llenguatge de consultes sobre el model de dades RDF i funciona sobre qualsevol font d'origen de dades que estigui mapejada sobre RDF.

L'especificació de SPARQL, que està en desenvolupament pel grup de recerca *RDF Data Access Working Group* del W3C, és un sistema independent que emula al llenguatge SQL per a bases de dades relacionals i XQuery per dades de XML.

SPARQL consisteix en tres especificacions: l'especificació del llenguatge de consultes (*query language specification*), el format de resultats XML (*query results XML format*) que descriu el format dels resultats de les consultes SPARQL, i el protocol d'accés a les dades remotes (*data access protocol*) que descriu el protocol per fer consultes remotes sobre bases de dades RDF.

SPARQL és a la vegada un llenguatge de consultes que especifica la sintaxi per la composició, concordància i experimentació, un protocol que descriu l'accés remot a dades i la transmissió de consultes dels clients als servidors. El resultats de les consultes es generen en XML i poden ser ordenats, limitats o presentats de diverses maneres.

Permet obtenir informació dels grafs RDF i proporcionar els següents serveis:

- Extreure informació en forma de URIs, nodes buits i literals.
- Extreure subgrafs RDF.
- Construir nous grafs RDF basats a partir dels grafs de dades.

Un **graf RDF** està format per un conjunt de **tripletes RDF** (subjecte–predicat–objecte). A continuació es defineixen els conceptes a partir dels quals es forma una consulta SPARQL:

- **subjecte** – format per un IRI o un node buit
- **predicat** – format per un IRI
- **objecte** – format per un IRI, un node buit o un literal

Un **node buit** s'escriu com `_:node` on node és el nom que se li vulgui donar.

Un **IRI** (*Inrternationalized Resource Identifiers*) és una generalització d'un URI que pot representar-se o bé delimitat entre < > o mitjançant un prefix més una etiqueta. Per exemple `<http://xmlns.com/foaf/0.1/>` o `foaf:name`.

Un **literal** RDF pot contenir una etiqueta per definir-ne una característica com per exemple @ per definir l'idioma o `^^` per definir el tipus. Per exemple `"chat"@fr` per definir l'idioma "fr" o `1^^xsd:integer` que definiria que `1` és de tipus `integer`.

[10] http://www.w3.org/TR/rdf-sparql-query/

Una **expressió RDF** és una part d'una tripleta RDF. Pot ser un IRI, un node buit o un literal. Per exemple `<uri>`, `_:b1` o `literal@en`.

Una **variable de consulta** és un identificador per combinar amb expressions RDF i es representa per `?` o `$` seguit pel nom que li volem donar, per exemple `?titol` o `$nom`.

Una **tripleta patró** és una tripleta que pot contenir variables com a subjecte, predicat i/o objecte separades per un espai i amb un punt al final. Un exemple és `?x foaf:name ?name .` (amb el punt final inclòs).

Un **graf patró** és un conjunt de tripletes patró. S'ha d'escriure entre `{ }` i entre cada tripleta hi ha d'haver un punt. N'hi ha de diferents tipus:

- **Bàsic** – graf patró que conté tripletes patró:
  `{ <http://exemple.com/abc ?y "Hola" .`
  `?subjecte $objecte "literal" }`
- **De grup** – graf patró que conté diversos grafs patrons que han de coincidir:
  { ?persona rdf:type foaf:Person
  { ?persona foaf:name "Aleix } }
- **Opcional** – graf patró que ha de fallar al coincidir i proporciona unions però no causa la fallida de la consulta. S'escriu amb la paraula clau `OPTIONAL`:
  `OPTIONAL { ?persona faof:nick ?nick }`
- **Unió** – una parella de grafs patró qualsevol dels quals ha de coincidir i unir les mateixes variables. S'escriu amb la paraula clau `UNION`:
  `{ ?node ex:name ?name } UNION { ?node vcard:FN ?name }`
- **Graf** – paraula clau per especificar el nom del graf per utilitzar o retornar el nom del graf com una unió. S'escriu amb la paraula clau `GRAPH` abans del graf patró:
  `GRAPH <http://exemple.com/myfoaf> {?persona foaf:name ?nom}`
  `GRAPH ?graph {?persona foaf:name ?nom}`

Una consulta SPARQL té la següent estructura:

Pròleg (opcional)	`BASE <IRI>` `PREFIX prefix: <IRI>`

Format del resultat	`SELECT (DISTINCT) ?variable1 ... ?variableN`
	`SELECT (DISTINCT) *`
	`DESCRIBE ?variable1 ... ?variableN`
	`DESCRIBE *`
	`CONSTRUCT {graf patró}`
	`ASK`

Conjunt de dades (opcional)	`FROM <IRI>` `FROM NAMED <IRI>`
Graf patró (opcional, obligatori per ASK)	`WHERE {graf patró [FILTER` *expressió*`]}`
Ordenació resultats (opcional)	`DISTINCT` `ORDER BY DESC/ASC`
Selecció resultats (opcional)	`LIMIT` *n*`, OFFSET` *m*

3.2. EXEMPLES DE CONSULTES

CONSULTA SIMPLE

En el primer exemple d'una consulta SPARQL senzilla per saber el títol del llibre prendrem el document RDF que conté aquesta sentència:

SUBJECTE	PREDICAT	OBJECTE
<http://example.org/book/book1>	<http://purl.org/dc/elements/1.1/title>	"SPARQL Tutorial"

L' estructura d'una consulta té en dues parts:

- **SELECT** - identifica les variables que volem que apareguin al resultat
- **WHERE** - conté una tripleta RDF.

Aquest seria el format de la consulta, en el que l'expressió `?title` representa la variable títol que volem que aparegui al resultat:

```
SELECT ?title
WHERE
{
  <http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title .
}
```

El resultat de la consulta anterior seria el següent:

title
"SPARQL Tutorial"

En el present document els resultats de les consultes es mostraran com una taula on cada columna correspondrà al conjunt de variables que es volen mostrar a la solució i cada fila correspondrà a cada solució.

CONSULTES AMB MULTIPLES RESULTATS

El següent exemple es mostra com una consulta mostra múltiples resultats mitjançant la utilització de l'ontologia *Dublin Core* (DC), que és una iniciativa per definir un estàndard de metadades per recuperar informació a través de diferents dominis del coneixement

Es defineix un prefix `@prefix dc: <http://purl.org/dc/elements/1.1/>` per abreujar les URIs i donar més claredat a les consultes.

Les dades són les següents:

SUBJECTE	PREDICAT	OBJECTE
_:a	dc:title	"SPARQL Query Language Tutorial"
_:a	dc:creator	"Alice"
_:b	dc:title	"SPARQL Protocol Tutorial"
_:b	dc:creator	"Bob"

L'expressió **_:a** representa un node buit. Fem una consulta per saber els títols i els seus creador:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?titol ?creador
WHERE
  { ?x dc:title ? title.
    ?x dc:creator ? creator }
```

En una consulta les variables del SELECT no se separen per comes sinó que es separen per un espai en blanc. Cal fer notar que en la documentació del W3C es comenta que el cost de modificar l'actual especificació per separar les variables de forma diferent arribaria a ser gran.

El resultat de la consulta retorna dues tuples:

title	Creator
"SPARQL Query Language Tutorial"	"Alice"
"SPARQL Protocol Tutorial"	"Bob"

CONSULTES D'EMPARELLAMENT DE LITERALS RDF

Els següents exemple es basaran sobre els següents literals RDF:

```
@prefix dt:   <http://example.org/datatype#>
@prefix ns:   <http://example.org/ns#>
@prefix :     <http://example.org/ns#>
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#>
```

SUBJECTE	PREDICAT	OBJECTE
:x	ns:p	"42"^^xsd:integer
:y	ns:p	"abc"^^dt:specialDatatype
:z	ns:p	"cat"@en

L'expessió **^^URI** defineix el tipus per a l'objecte indicat en la URI.

```
SELECT ?v WHERE { ?v ?p 42 }
```

En la consulta es demana el subjecte que té com a valor l'objecte "42".

v
<http://example.org/ns#x>

La solució és el subjecte `<http://example.org/ns#x>` que és el mateix que `:x` ja que el prefix buit està definit per `@prefix : <http://example.org/ns#>`.

En la següent consulta la solució és buida ja que el tipus `"cat"` no és el mateix literal RDF que `"cat"@en` del subjecte `:z` :

```
SELECT ?x WHERE { ?x ?p "cat" }
```

CONSULTES DE RESTRICCIONS DE VALORS

Considerarem les següents dades:

```
@prefix dc:   <http://purl.org/dc/elements/1.1/> .
@prefix :     <http://example.org/book/> .
@prefix ns:   <http://example.org/ns#> .
```

SUBJECTE	PREDICAT	OBJECTE
:book1	dc:title	"SPARQL Tutorial"
:book1	ns:price	42
:book2	dc:title	"The Semantic Web"
:book2	ns:price	23

Uns consulta pot ser restringida per comparar un `string` amb una expressió regular utilitzant l'operador `regex` :

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   { ?x dc:title ?title
          FILTER regex(?title, "SPARQL")
        }
```

Aquesta consulta en donaria com a resultat el títol del subjecte `book2`:

title
"SPARQL Tutorial"

Les comparacions de les expressions regulars poden no tenir en compte la diferenciació entre majúscules i minúscules (case-insensitive) amb el flag "`i`" :

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   { ?x dc:title ?title
          FILTER regex(?title, "web", "i" )
```

title
"The Semantic Web"

També és possible restringir valors dels literals que tenen valors numèrics. Els filtres no contemplen la seva forma lèxica sinó que apliquen el valor del literal:

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price
WHERE   { ?x ns:price ?price .
          FILTER (?price < 30.5) .
          ?x dc:title ?title . }
```

Restringint la variable del preu, el filtre de la consulta pregunta si el subjecte ?x té un preu menor de 30,5 i només el subjecte book2 compleix aquesta condició:

title	price
"The Semantic Web"	23

CONSULTES ALTERNATIVES

L'operador `UNION` permet realitzar consultes sobre condicions alternatives.

Exemple de consulta utilitzant `UNION`:

```
@prefix dc10:  <http://purl.org/dc/elements/1.0/>
@prefix dc11:  <http://purl.org/dc/elements/1.1/>
```

SUBJECTE	PREDICAT	OBJECTE
_:a	dc10:title	"SPARQL Query Language Tutorial"
_:a	dc10:creator	"Alice"
_:b	dc11:title	"SPARQL Protocol Tutorial"
_:b	dc11:creator	"Bob"
_:c	dc10:title	"SPARQL"
_:c	dc11:title	"SPARQL (updated)"

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>

SELECT ?title
WHERE  { { ?book dc10:title  ?title } UNION { ?book dc11:title  ?title } }
```

Els resultats de la consulta són tots els títols dels subjectes que hi ha a les dades:

Title
"SPARQL Protocol Tutorial"
"SPARQL"
"SPARQL (updated)"
"SPARQL Query Language Tutorial"

Si es vol especificar de quin predicat (dc10 o dc11) és cada títol es faria la següent consulta:

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>

SELECT ?x ?y
WHERE  { { ?book dc10:title ?x } UNION { ?book dc11:title  ?y } }
```

En el resultat de la consulta es distingiran entre els resultats d'un predicat o de l'altre en diferents columnes:

x	y
	"SPARQL (updated)"
	"SPARQL Protocol Tutorial"
"SPARQL"	
"SPARQL Query Language Tutorial"	

En aquest apartat s'ha vist les característiques del llenguatge SPARQL i el seu format per realitzar consultes sobre dades RDF. En el següent s'estudiaran les bases de dades amb suport a dades RDF i les seves característiques. També s'avaluaran les funcionalitats i es veurà l'arquitectura de 3 bases de dades del mercat que tenen suport a aquest tipus de format de dades.

4. ANÀLISI DELS SGBD AMB SUPORT AL LLEGUANTGE SPARQL

4.1. INTRODUCCIÓ A LES BASES DE DADES XML

És conegut que les bases de dades són una part fonamental de totes les organitzacions ja que en elles es guarda informació per al desenvolupament de les seves activitats. L'aparició d'XML ha revolucionat el present i futur de l'administració de les bases de dades doncs aquest llenguatge ha permès trencar barreres i crear una manera estàndard de processar la informació.

XML està provocant l'aparició de noves tecnologies, entre elles, l'aparició de les **bases de dades XML natives**, una nova generació de bases de dades que si bé es troben en fase d'investigació i desenvolupament, en un futur poden ser una bona alternativa a les ja conegudes bases de dades relacionals.

Actualment les bases de dades relacionals ja donen suport per a XML però segueixen emmagatzemant la informació de manera relacional (en forma tabular: taules, registres, columnes) o emmagatzemen tot el document en format BLOB (*Binary Language Object*). Tampoc estan preparades per guardar estructures de tipus jeràrquic, com són els documents XML o RDF, pels següent motius:

- Les BD relacionals tenen una estructura regular homogènia mentre que els documents XML tenen caràcter heterogeni.

- Els documents XML solen contenir molts nivells d'aniuament mentre que les dades relacionals són «planes» (tenen un nombre de columnes per cada fila).

- Els documents XML tenen un ordre intern mentre que les dades relacionals no estan ordenades.

- Les dades relacionals generalment són denses (cada columna té un valor i acostumen a tenir pocs valors nuls) mentre que els documents XML són dispersos ja que poden presentar manca d'informació mitjançant l'absència d'un l'element.

S'han desenvolupat teories per fer *encaixar* objectes i estructures jerarquitzades a bases de dades relacionals, com per exemple les regles de transformació relacionals en DTD (*Document Type Definition*) que es detallen a continuació:

1. Per cada taula en l'esquema de la BD cal crear un element amb el mateix nom de la taula i la cardinalitat.

2. Les columnes de la taula estan incloses en un altre element, es a dir, un sub-element de l'element creat mitjançant la regla anterior, que representa un registre de la taula.

3. Per cada columna de la taula que el seu tipus de dades és simple (tipus `char`, `int`, etc.) cal crear un element, sub-element de l'element creat en la regla 2, del mateix tipus.

4. Per cada columna de la taula que el seu tipus de dades és complexe (tipus objecte) cal crear un element complex, subelement de l'element creat en la regla 2, amb el mateix nom de la columna. Per a cada propietat del tipus d'objecte caldrà crear un element amb el mateix nom de la propietat.

5. Per cada columna de la taula que és una taula aniuada, cal crear un element amb el mateix nom d'aquella columna i la cardinalitat adequada. També caldrà repetir els passos a partir de la regla 2.

Les bases de dades habilitades per a XML suporten XML com a format d'entrada/sortida, es a dir, poden traduir els seus formats de dades internes a XML i viceversa. Aquest tipus de bases de dades permeten intercanviar dades més fàcilment amb altres que plataformes i poden programar-se mitjançant codi escrit segons les especificacions XML. Tot plegat ha produït una àmplia utilització d'XML per connectar els diferents sistemes existents dins una empresa amb altres sistemes de la mateixa empresa, amb sistemes dels seus clients o proveïdors, i per presentar dades *online* als consumidors via Internet.

Els principals avantatges i inconvenients d'emmagatzemar dades XML en una base de dades relacional són els següents:

- Els SGBD relacionals fa temps que estan al mercat i per tant estan àmpliament utilitats i provats.

- Possibilitat de ser utilitzades des d'aplicacions existents.

- La conversió és senzilla si les dades es generen a partir d'un esquema relacional i s'utilitza XML com format d'intercanvi de dades. Si XML no es genera a partir d'un esquema relacional la transformació no és tant senzilla. Aquest problema de la conversió es produeix especialment en elements aniuats i elements amb atributs multivalor.

Enfront de les bases de dades relacionals hi ha un altre tipus de bases de dades, les bases de dades XML natives, que suporten XML en les seves arquitectures internes i ofereixen avantatges significatius respecte de les relacionals. Les XML:BD natives són més escalables i realment més interoperables que les bases de dades que només utilitzen XML com un format per l'intercanvi de dades.

L'organització *XML:DB Initiatives for XML Databases*[11] defineix una base de dades nativa en XML com un model lògic per a documents XML que permet emmagatzemar i recuperar document d'acord amb aquest model, i per tant tenen la capacitat d'obtenir resultats de les consultes en format XML.

Mentre que les BD relacionals són centrades a dades (*data-centric databases*) ja que guarden dades atòmiques, les BD natives són centrades a document (*document-centric databases*) ja que una XML:DB emmagatzema documents XML i no conté camps ni dades atòmiques.

[11] http://sourceforge.net/projects/xmldb-org/

Tot i que cada proveïdor té característiques diferents, en general totes les XML:BD natives tenen les següents característiques comuns:

Emmagatzematge
Les BD natives guarden els documents creant models lògics. Aquests models es basen directament en XML o en alguna de les tecnologies relacionades com DOM[12] (*Document Object Model*) o InfoSet[13]. Aquests models inclouen diferents nivells d'agregació i complexitat així com un suport complert per a la gestió de dades semiestructurades. Els models són automàticament mapejats per les bases de dades al mecanisme d'emmagatzematge corresponent. El mapeig usat per aquests bases de dades garantirà la correcta utilització del model associat al document original.

Col·leccions
Aquests tipus de bases de dades treballa sobre col·leccions de documents cosa que permet manipular aquests documents com un sistema. És similar al concepte relacional d'una taula tot i que es diferencia en el sentit en que no requereix que la col·lecció tingui un esquema associat. Això significa que es pot emmagatzemar qualsevol document XML en una col·lecció sense necessitat de cap esquema i tot i això es poden fer consultes a través de tots els documents de la mateixa col·lecció. Aquesta propietat s'anomena esquema–independent.

Processament de dades
El processament de dades en aquests tipus de bases de dades sembla ser molt àgil però no és així degut al format jeràrquic en el que està emmagatzemada la informació. Moltes BD necessiten que es recuperi tot el document XML, i que s'actualitzi amb l'API XML que s'estigui fent servir. Posteriorment cal que es torni a guardar al repositori. Això és degut a que no existeix un llenguatge estàndard que permeti l'actualització, inserció o eliminació d'una manera senzilla i ràpida. Actualment s'està desenvolupant el llenguatge XUpdate[14] per fer aquesta tasca, però moltes XML:BD encara no el suporten.

Consultes
Aquests tipus de bases de dades no utilitzen SQL com a llenguatge de consulta; enlloc d'aquest utilitzen XPath[15] o XQuery[16]. Algunes BD permeten seleccionar els elements que hauran de tenir índexs mentre que d'altres indexaran tot el contingut del document. El problema que tenen és que només permeten realitzar actualitzacions en un document XML (amb el llenguatge XUpdate) malgrat que alguns gestors d'aquests tipus de bases de dades no ho suporten.

Des de que l'any 2004 el W3C va convertir RDF en recomanació per desenvolupar la Web Semàntica també ha sorgit la necessitat de desenvolupar bases de dades natives en RDF. A l'igual que ha succeït amb les XML:BD natives, les BD relacionals s'han desenvolupat per donar suport a RDF però també estan apareixent RDF:BD natives, tal com veurem en el següent apartat.

[12] http://www.w3.org/TR/REC-DOM-Level-1/
[13] http://www.w3.org/TR/xml-infoset/
[14] http://xmldb-org.sourceforge.net/xupdate/
[15] http://www.w3.org/TR/xpath
[16] http://www.w3.org/TR/xquery/

4.2. BASES DE DADES AMB SUPORT A RDF

En aquests moment hi ha un gran nombre de BD que permeten treballar amb dades RDF: Jena2[17], Sesame[18], RDF Gateway[19] , Oracle10g[20], Kowari[21] o rdfBD[22], entre d'altres. Molts dels sistemes actuals implementen l'emmagatzematge persistent mitjançant BD relacionals on les dades es guarden a taules relacionals on:

- Les tripletes es guarden a una taula de sentències amb referències a les taules de suport.

- La taula de sentències té les columnes subjecte, predicat i objecte, i cada fila representa una tripleta

L'emmagatzematge a una BD relacional requereix un disseny amb poques taules (amb pocs atributs i moltes files). Cada taula conté d'una a cinc columnes, la majoria indexades més d'un cop per aconseguir un major rendiment. Una consulta bàsica requereix que totes les taules siguin unides amb altres per realitzar *joins* aniuats. Poden ser necessàries una gran quantitat de consultes individuals per realitzar consultes més complexes produint-se més temps de resposta per aquestes consultes.

L'actualització d'una base de dades relacional que desa informació en RDF és costosa ja que totes les columnes estan indexades. Per a fer-ho es necessita que les pàgines de dades i les columnes indexades es bloquegin.

Els optimitzadors de consultes de bases de dades relacionals no funcionen gaire bé amb taules amb moltes files i pocs atributs, amb gran variació en la distribució de dades. Això succeeix per que les columnes que guarden una gran varietat de dades estan dissoltes respecte a les columnes amb una unicitat. A més a més alguns optimitzadors de RDBMS (*Relational Data Base Management System*) estan dissenyats per examinar tots els possibles plans d'execució basats en regles i costos. Donat que les consultes de metadades tendeixen a ser recursives i tenen diversos camins, als optimitzadors no els resulta senzill determinar el millor pla d'execució, i fins i tot, en alguns casos, determinar el pla d'execució comporta més temps que executar la pròpia consulta.

Un altre problema derivat d'emmagatzemar les dades RDF en bases de dades relacionals és la seguretat. Es necessiten taules addicionals en el disseny per garantir les dades a nivell de taula i columna. Això provoca que incrementi el numero de *joins* realitzats i la complexitat de les consultes.

[17] http://jena.sourceforge.net/
[18] http://www.openrdf.org/
[19] http://www.intellidimension.com/
[20] http://www.oracle.com/
[21] http://www.kowari.org/
[22] http://www.guha.com/rdfdb/

4.3. AVALUACIÓ DE FUNCIONALITATS

Per detallar les funcionalitats dels SGBD a continuació es mostraran diferents productes existents en el mercat dels que es veurà l'arquitectura i les seves funcionalitats.

4.3.1. Jena2

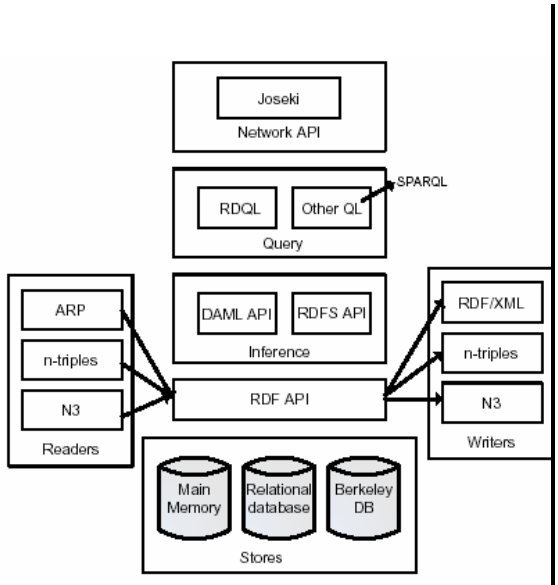
Jena és una eina Java per manipular models RDF. Proporciona un entorn de programació per a RDF, RDF-Schema i OWL, incloent un motor d'inferència basat en regles. El codi de Jena és lliure i ha estat desenvolupat pel *HP Labs Semantic Web Research*.

El *framework* de Jena2 inclou:

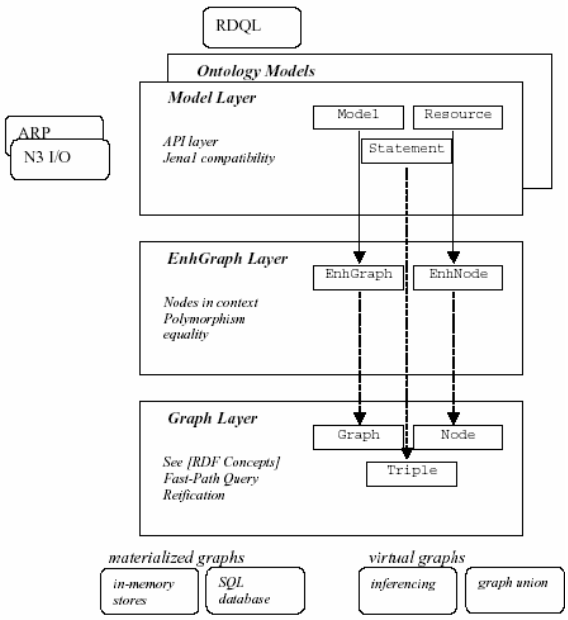
- Una API (*Application Programming Interface*) per RDF .
- Una API d'ontologies que permet treballar amb OWL, DAML i RDFS.
- Un analitzador sintàctic per RDF.
- Un motor d'inferència.
- El motor de consultes ARQ (que inclou SPARQL).
- Lectura i escriptura de RDF en format RDF/XML, N3 i N-Tripes.
- Un sistema d'emmagatzematge a memòria i persistent.

Els enunciats del model es guarden en un graf base. El motor d'inferència pot utilitzar el contingut d'aquest graf amb les regles semàntiques del llenguatge per mostrar un conjunt complert d'enunciats. Això implica incloure també aquells enunciats que es poden inferir a partir del graf base. La interfície *Graph* pot ser utilitzada per accedir a les dades del model.

L'arquitectura de Jena es pot veure en els següents esquemes:



Arquitectura de Jena



Arquitectura de Jena2

El nucli principal de l'arquitectura és l'API RDF que suporta la creació, manipulació i consulta de grafs RDF. La API suporta tecnologies d'emmagatzematge. El *plug-in interfaces* permet lectura i escriptura automàtica per diferents llenguatges que es poden utilitzar per representar grafs RDF. En aquesta vista és convenient manipular els grafs RDF com un tot.

Jena també inclou ARP, un *parser* que llegeix RDF/MXL, el llenguatge estàndard per la representació de grafs RDF. Enlloc de tenir un *parser* codificat a mà, a Jena el compilador de Java (`Javacc`) genera el *parser* ARP a partir de la gramàtica RDF/XML. ARP utilitza un *parser* XML estàndard com preprocessador lèxic i el *parser* generat és el que processa els símbols.

Joseki[23] és un servidor SPARQL de Jena per publicar models RDF a la web. Suporta el protocol SPARQL i també el llenguatge de consultes SPARQL. Es proporciona juntament amb la instal·lació de Jena i proporciona: dades RDF d'arxius o bases de dades, implementació de HTTP (GET i POST) pel protocol SPARQL, implementació de SOAP (*Simple Object Access Protocol*) pel protocol SPARQL i consultes *online* amb SPARQL.

Jena2 permet tres implementacions de l'API per emmagatzemar dades: a memòria, a una BD relacional o incrustades a la base de dades Berkeley DB[24]. La implementació a BD relacionals utilitza qualsevol base de dades que suporti JDBC.

ARQ és un motor de consultes per Jena que té les següents funcionalitats:

- Fer consultes en diferents llenguatges (SPARQL, RDQL o ARQ, el llenguatge de consultes experimental de la pròpia aplicació).
- Múltiples motors de consultes (de propòsit general o d'accés remot)
- Traducció de les consultes a SQL

Respecte al model físic, la segona versió de Jena millora alguns problemes de la primera versió: el disseny de l'esquema s'ha modificat per millorar problemes de rendiment deguts a excessives unions de taules i l'increment de l'emmagatzematge resultant de la reificació.

Utilitza les tripletes d'una forma desnormalitzada (en la primera versió ho feia de forma normalitzada), multi-model i de triple emmagatzematge. Els models es guarden en taules separades i cada model conté les sentencies afirmades en una taula i les sentencies reificades en una altra, encara que el comportament per defecte pot ser modificat segons la configuració que es defineixi. La taula de sentències confirmades guarda els valors textuals reals de les tripletes en columnes de subjecte-predicat-objecte. A més a més els valors textuals es guarden redundantment si el mateix valor textual apareix en diferents tripletes. Aquest fet fa que es consumeixi més espai d'emmagatzematge que en la primera versió però aporta un millor rendiment en el temps de resposta d'una consulta ja que el nombre de *joins* de les taules requerides es redueix.

[23] http://www.joseki.org/
[24] http://www.oracle.com/database/berkeley-db.html

Les sentencies reificades es guarden en una taula de propietats que conté les columnes StmtURI (que representen la sentència reificada), rdf:subject, rdf:predicate, rdf:object i rdf:type. Una fila de la taula amb tots els atributs presents representa una tripleta reificada.

A banda de les taules per sentències afirmatives i reificades, Jena2 es pot configurar per incloure taules de propietats en la creació de grafs. Aquestes taules guarden parells de subjecte-valor per predicats específics. La taula columna inclou el subjecte i cada predicat que serà representat a la taula. Aquestes taules estan dissenyades per proporcionar una lleugera reducció de l'emmagatzematge ja que els predicats URIs no es guarden. S'intenta agrupar propietats que són accedides conjuntament i d'aquesta manera incrementar el rendiment.

Aquestes són les taules bàsiques que componen l'esquema de Jena2 però n'hi ha d'altres de suport. Per tant Jena2, com d'altres sistemes que implementen emmagatzematge persistent en bases de dades relacionals, utilitza taules relacionals planes.

El mòdul de base de dades de Jena proporciona una implementació de la interfície però amb capacitat per guardar i recuperar dades RDF utilitzant una base de dades. Actualment suporta MySQL, Oracle i PostgreeSQL per emmagatzemar persistentment, i s'executa sota Windows y Windows XP.

El subsistema de persistència suporta la capacitat *Fastpath* per a consultes RDQL, que genera consultes SQL dinàmicament per millorar tant com sigui possible el rendiment de les consultes RDQL en el motor de la base de dades.

L'algoritme *Fastpath* treballa dividint les variables i les constants d'una consulta en un o més grups – anomenats *stages* – on cada *stage* consisteixen en variables que poden ser processades juntes produint-se els dos casos següents, considerats més eficients que realitzar una consulta SQL:

- Els *stages* que consisteixen en més d'una variable són processats mitjançant una consulta simple generada dinàmicament.

- Els *stages* que consisteixen en només una variable són processades mitjançant l'operació *Find*.

L'algoritme *Fastpath* també determina l'ordre de l'execució per intentar minimitzar el temps d'execució.

4.3.2. Sesame

Sesame proporciona una arquitectura genèrica, implementada amb codi obert en Java, que permet l'emmagatzemament persistent de dades i esquemes RDF en una BD relacional o en memòria, i la seva posterior consulta *online*.

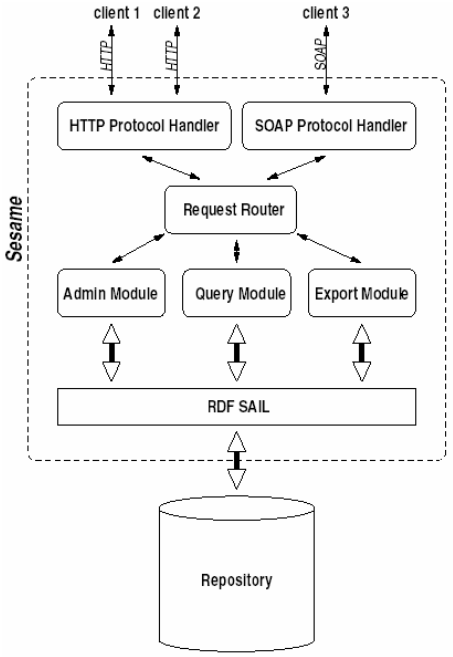
Suporta els llenguatges de consultes RQL, RDQL i SeRQL (el llenguatge de consultes pròpi). En la versió Sesame 2.0 - alpha4 (publicada el novembre de 2006) s'ha redissenyat el motor de consultes per poder suportar consultes amb SPARQL.

Sesame depèn d'una BD relacional per guardar les dades RDF. Actualment suporta PostgreSQL[25], MySQL[26], Oracle9i[27] i SQL Server[28]. Per treballar només amb memòria no cal instal·lar el SGBD.

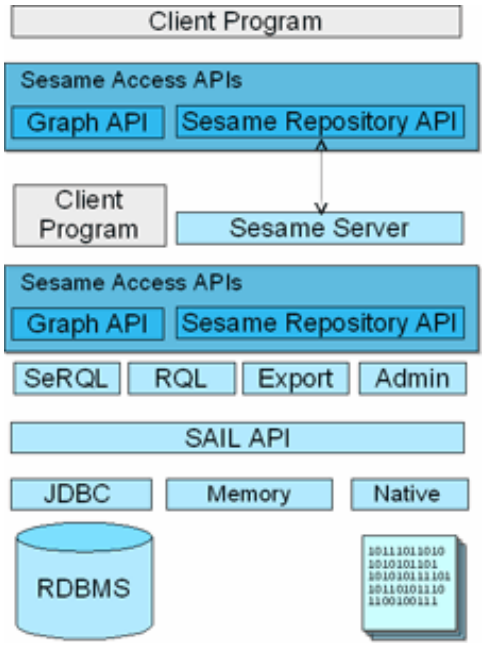
Un dels seus principals components de l'arquitectura de Sesame és el SAIL (*Storage and Interface Layer*) que és el nivell que s'encarrega d'interactuar amb els SGBD suportats permetent mantenir l'arquitectura de Sesame independent d'un SGBD concret. SAIL és un API que proporciona mètodes RDF específics als seus clients i els tradueix a crides d'un SGBD específic. Els mòduls funcionals són clients de SAIL i actualment n'existeixen tres:

- El motor de consultes RQL
- El mòdul d'administració RDF
- El mòdul d'exportació RDF

L'arquitectura de Sesame es pot veure en els següents esquemes:



Arquitectura de Sesame

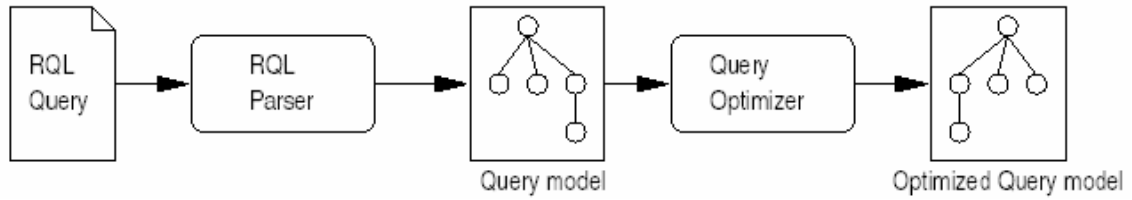


Arquitectura ampliada de Sesame

[25] http://www.postgresql.org/
[26] http://www.mysql.com/
[27] http://www.oracle.com/
[28] http://www.microsoft.com/sql/

Depenent de l'ambient en el que Sesame s'implanti ofereix diverses formes de comunicació: HTTP per a contextos web, RMI (*Remote Method Invocation*) o SOAP (*Simple Object Access Protocol*).

El mòdul de consultes RQL:
Malgrat obeir els estàndards del W3C, no suporta la tipificació de dades. Tal com es pot veure en el següent esquema, en el procés de consultes se segueixen els següents passos: després del *parsing* i la construcció de l'arbre de consultes, es passa a un optimitzador de consultes que transforma la consulta en un model equivalent però més eficient:



L'optimització consta d'un conjunt d'heurístiques (pre-avaluacions que no depenen de cap mètode d'emmagatzematge concret).

El model optimitzat és avaluat seguint l'estructura d'arbre generada, cada objecte representa una unitat bàsica de consulta original i s'avalua a si mateixa extraient informació per mitjà de SAIL quan la necessita. El gran avantatge d'aquest enfocament és que els resultats es retornen de la mateixa manera, enlloc de construir primer la consulta complerta a memòria.

El gruix de les consultes són resoltes pel motor RQL, per exemple quan una consulta conté una operació *semi-disjoin* on cada sub-consulta es avaluada i l'operació de *semi-disjoin* s'executa llavors, pel motor de consultes, sobre els resultats.

Una altra alternativa seria aprofitar l'optimització a partir de l'SGBD, però això faria que el procés de consultes fos dependent d'un SGBD concret.

El mòdul d'administració:
Per fer possible la inserció de dades RDF i informació d'esquemes en un repositori, Sesame utilitza el módul d'administració oferint dues operacions fonamentals:

- Addició incremental de dades i esquemes RDF.
- Netejar un repositori.

Cal destacar que una opció d'eliminació funcional encara no està disponible.

El mòdul extreu la informació d'una font RDF/RDFS, generalment en XML serialitzat, i es fa un *parsing* utilitzant un *parser* RDF (per exemple, l'ARP RDF *parser* que és part del paquet de Jena), lliura la informació al mòdul d'administració de la forma bàsica (objecte, atribut, valor) i després al repositori, mitjançant la capa SAIL, reportant els errors que hagin pogut succeir.

El mòdul d'exportació RDF:
Aquest mòdul permet exportar els continguts dels repositoris en format XML serialitzat. L'objectiu d'aquest mòdul és combinar Sesame amb altres eines RDF donat que la majoria reconeixen el format. També permet l'exportació de dades, esquemes, o els dos en funció de si és necessaria la semàntica o no.

SAIL:
La SAIL API consta d'un conjunt d'interfícies Java que han estat específicament dissenyades per l'emmagatzematge i extracció d'informació basada en RDFS. Els principis més rellevants de SAIL són els següents:

- Defineix una interfície bàsica per emmagatzemar, esborrar i recuperar RDF i RDFS de repositoris existents.

- Proporciona una abstracció del mecanisme actual d'emmagatzematge sent aplicable a SGBD relacionals, sistemes d'arxius, emmagatzematge en memòria...

- És aplicable a diferents plataformes *hardware* i *software*, com PDAs, però ofereix suficient llibertat per l'optimització en el cas de gran quantitat d'informació, per exemple *clusters* de BD a nivell d'empreses.

- És extensible a altres llenguatges basats en RDF com DAML+OIL.

Una característica important que diferencia a l'API SAIL, per exemple, de Jena, és que ofereix mètodes per a consultar classes i propietats, així com restriccions de domini i de rang. Això no passa amb altres productes que estan enfocats al conjunt de tripletes deixant el treball de la interpretació a l'usuari. Aquest avantatge es basa en la forta relació entre l'eficiència de la inferència i el model d'emmagatzematge utilitzat.

Una altra característica de SAIL es relaciona al seu tamany «lleuger» donat que només implementa quatre interfícies predefinides, oferint emmagatzematge, recuperació, funcionalitat i suport transaccionals.

4.3.3. RDF Gateway

RDF Gateway és una plataforma dissenyada per la Web Semàntica. És a la vegada un servidor d'aplicacions, un servidor web i un servidor de bases de dades RDF. El seu motor de base de dades no és com els SGBD relacionals, sinó que suporta un llenguatge de comandes similar a SQL, múltiples BD, taules emmagatzemades a memòria o disc, cursors i transaccions.

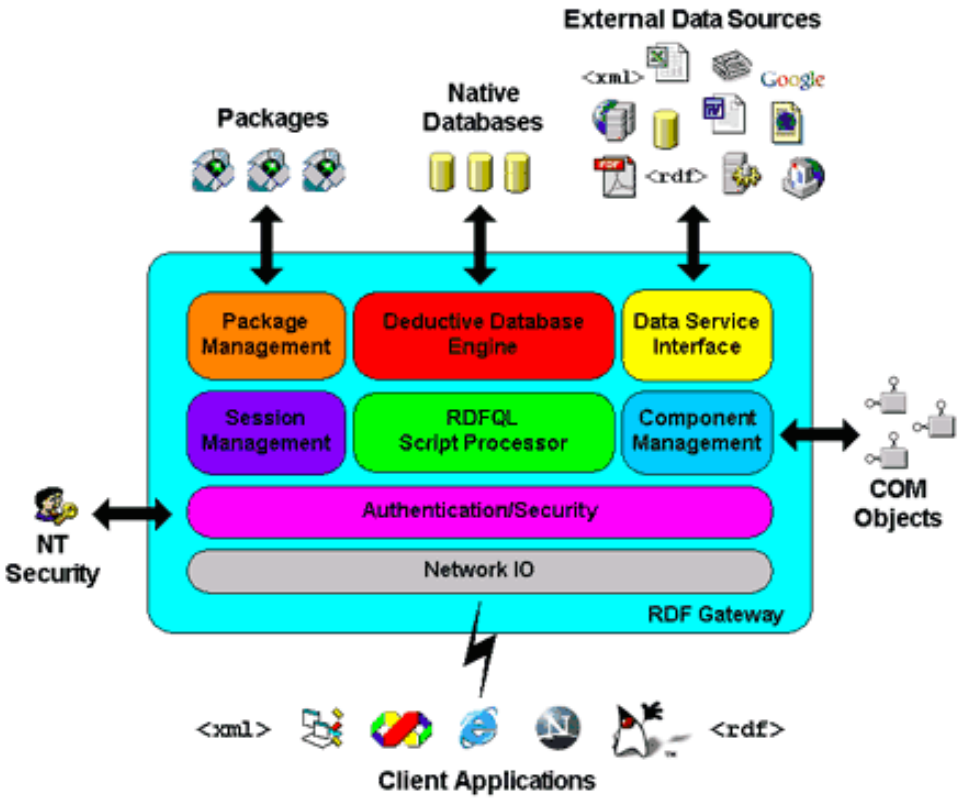
La principal diferència entre RDF Gateway i una BD relacional és com es representa la semàntica d'un model de dades. Una BD relacional codifica la semàntica implícitament utilitzant elements d'emmagatzematge com taules, columnes i claus foranes. Les aplicacions client han de ser desenvolupades per comprendre la semàntica de la informació relacional. Això fa que sigui difícil compartir informació entre sistemes que no comparteixen un esquema relacional idèntic. RDF Gateway no necessita codificar la semàntica: totes les taules tenen el mateix format i les claus foranes no s'utilitzen. Donat que qualsevol model pot ser descrit utilitzant tripletes RDF, totes les taules tenen el mateix format amb quatre columnes: context (que pot emmagatzemar l'URI), predicat, subjecte i objecte.

Amb RDF Gateway les taules no s'associen amb una classe simple d'objecte al model de dades ja que no es poden guardar objectes de moltes classes diferents. Una taula fins i tot pot mantenir taules de diversos esquemes diferents. Per tant, les taules de RDF no són més que fonts de dades.

Algunes de les seves característiques sobre la gestió de les dades RDF són les següents:

- Utilitza comandes similars a SQL per realitzar consultes a dades RDF.
- Raonament deductiu mitjançant regles d'inferència (llibreries per OWL i RDFS).
- Alt rendiment en disc i taules de memòria.
- Transaccions amb nivells d'aniuament.
- Taules virtuals que integren fonts de dades externes (pàgines web, documents locals, BD relacionals, entre d'altres).
- Consultes federades a través de múltiples fonts de dades internes i externes.
- Cerques de text complert.
- Encriptació de dades.
- Suport per la serialització per RDF/XML, N3 i NTriples.
- Seguretat a nivell de taules i sentències.
- Accés a dades des d'altres aplicacions via HTTP, ADO (*ActiveX Data Objects* de Microsoft) o JDBC (*Java Database Connectivity* de Sun).

El següent esquema representa l'arquitectura de RDF Gateway:



Processador d'*Scripts* RDFQL[29]: és una màquina virtual que compila i executa els *scripts* d'RDFQL. RDFQL és un llenguatge *script* orientat a servidors que integra extensions de consultes similars a SQL per proporcionar accés al motor de base de dades deductiu d'RDF Gateway. També permet que les pàgines continguin una combinació entre *script* i contingut estàtic similar a ASP (*Active Server Parges* de Microsoft).

Base de dades deductiva: conté un motor de base de dades que ha estat dissenyada des del principi per suportar RDF. El seu disseny semi-estructurat és ideal per consultar i emmagatzemar tripletes RDF. El motor deductiu de la BD realitza una avaluació de les consultes que és federada durant totes les fonts de dades especificades. Les capacitats d'inferència lògica del motor proporcionen suport per la sintaxi declarativa d'RDFQL. Implementa la seva pròpia persistència en arxius, eliminant la necessitat d'un sistema d'emmagatzematge extern.

Interfície de servei de dades: un *Data Service Interface* és un mòdul que implementa una interfície que mostra els continguts d'una font de dades, com per exemple dades RDF. Per tant, la interfície de servei de dades permet l'accés a fonts de dades extenses i que es puguin integrar dins de l'RDF Gateway. RDFQL permet que es realitzin consultes federades sobre múltiples serveis de dades. Aquesta interfície oberta fa possible utilitzar qualsevol proveïdor de servei de dades disponible en l'actualitat o desenvolupar-ne un personalitzat per una font de dades.

Autenticació/Seguretat: disposa d'un model de seguretat basat en drets i permisos per controlar l'accés a recursos del servidor i de la BD.

E/S de xarxa: conté una interfície de xarxa optimitzada per pel rendiment i la escalabilitat. Suporta tant HTTP com un protocol propietari basat en TCP/IP. La capa de xarxa E/S suporta esquemes segurs d'autenticació de xarxa com el protocol NTLM (*Windows NT LAN Manager*) d'autenticació de xarxa. RFG Gateway té *drivers* clients per ADO i JDBC. Això li permet suportar gran quantitat de clients com navegadors web, aplicacions Windows o Java i clients basats en XML o RDF.

Gestió de paquets: *Package Management* permet que les aplicacions es desenvolupin i distribueixin com paquets. Un paquet consisteix en pàgines RDF Server, pàgines HTML, imatges o qualsevol tipus d'arxiu.

Gestió de components: RDFQL suporta COM en els seus *scripts* al servidor. Això permet que la funcionalitat de RDF s'estengui per que les aplicacions s'integrin amb RDF Gateway.

[29] La versió 2.3.3 d'abril del 2006, entre d'altres modificacions i millores, ha afegit suport per al llenguatge SPARQL i modificat la interfície de servei de dades SQL per millorar l'eficiència amb MySQL.

4.4. ANÀLISI COMPARATIU

Després d'haver vist en l'apartat anterior les característiques de tres bases de dades amb suport a RDF, a continuació es fa un anàlisi comparatiu. El criteris han estat el tipus d'emmagatzematge de la BD, els llenguatges de consulta sobre la base de dades suportats, el motor de consultes utilitzat pel SGBD, les ontologies acceptades, els formats de lectura/escriptura a la BD, la escalabilitat, la possibilitat de l'SGBD de poder fer inferència i la possibilitat de la BD de representar les dades mitjançant una graf RDF.

En el següent quadre es mostra una comparativa de les principals diferències entre les tres BD:

		Jena	**Sesame**	**RDF Gateway**
	Versió	2.4	1.2.6	2.3.3
	Empresa	HP Labs	Aduna B.V.	Intellidimension, Inc.
	Llicència	Open Source	Open Source	Comercial
	Implementació	Java	Java	-
	Plataforma	Windows 98, XP		Windows NT, 2000, XP
EMMAGATZEMATGE	Natiu	NO	SI	SI
	Oracle	SI	SI	NO
	MySQL	SI	SI	NO
	PostgreSQL	SI	SI	NO
	SQL Server	Si	SI	NO
LLENGUATGES CONSULTA	SPARQL	SI	SI (versió 2.0)	SI
	RDQL	SI	SI	NO
	RQL	NO	SI	NO
	Propi	(RDQL)	SeRQL	RDFQL
MOTOR CONSULTES		sobre ARQ (pròpi)	sobre RQL	sobre RDFQL (pròpi)
ONTOLOGIES	OWL	SI	SI (extensible)	NO
	DAML+OIL	SI	SI	NO
	RDF-Schema	SI	SI	NO
FORMATS LECTURA/ ESCRIPTURA	RDF/MXL	SI	SI	SI
	N-Triple	SI	SI	SI
	N3	SI	NO	SI
	ARP	SI	SI	NO
ESCALABILITAT		SI	Moduls Java	ActiveX, DLL
INFERÈNCIA		SI	SI	SI
SERVIDOR WEB		SI (Joseki)	SI	SI
INTERFÍCIES CLIENTS	Java RMI	NO	SI	NO
	HTTP	SI	SI	SI
	SOAP	SI	SI	NO
	ADO	NO	NO	SI
	JDBC	SI	SI	SI
	SPARQL(protocol)	SI	SI (versió 2.0)	NO
	RDFCLI	NO	NO	SI
GRAFS RDF		SI	SI (versió 2.0)	NO

Sobre la taula anterior cal comentar que Jena i Sesame disposen de més opcions que RDF Gateway, que només suporta un emmagatzematge nadiu mentre que Jena i Sesame han implementat accés a la majoria de les plataformes amb les que s'està treballant actualment.

Un altre aspecte a destacar és que Sesame suporta tots els llenguatges de consulta (la recent versió 2.0 ha incorporat la implementació de SPARQL i només manté el llenguatge propi SeRQL de l'anterior versió).

Tots tres DBMS disposen de motor de consultes però RDF Gateway no proporciona suport per a ontologies, mentre que els altres dos les suporten totes, el que li suposa un clar desavantatge de cara a un futur per a contribuir dins el projecte de la Web Semàntica.

En quant als formats de lectura/escriptura de dades RDF tots tres els suporten gairabé tots (Jena és l'únic que els implementa tots).

Per últim, sobre les interfícies clients tots tres n'implementen varies de les que Sesame n'implementa una més que les altres dues plataformes.

En resum, RDF Gateaway està quedant obsolet respecte als altres dos gestors de bases de dades RDF estudiats. Els principals inconvenients que té són que no implementa accés a cap base de dades que no sigui la pròpia, que no suporta ontologies i que no suporta grafs RDF.

Jena i Sesame són bastant semblants en el sentit que tots dos implementen la gran majoria de tecnologies estudiades i l'únic fet diferencial que es podria comentar és que a diferencia de Jena, Sesame disposa d'independència dels DBMS degut a la seva arquitectura.

5. PROTOTIPUS

5.1. INTRODUCCIÓ

Prèviament a la realització del disseny del prototipus per realitzar consultes SPARQL sobre dades RDF cal decidir l'arquitectura i tecnologies que s'utilitzaran. Després de la fase d'anàlisi feta en l'apartat anterior i la comparativa entre les diverses aplicacions estudiades s'ha decidit desenvolupar aquest prototipus utilitzant Sesame en la seva versió 2.0 alpha 4 publicada el novembre de 2006.

Els factors que han fet decidir la utilització d'aquest *framework* són la independència de Sesame de cap DBMS i la simplicitat que suposa poder crear un repositori a memòria principal sense requerir de la instal·lació de cap servidor de base de dades.

La recent aparició d'aquesta versió també ha motivat la seva utilització donat que al tractar-se d'una eina de tipus *opersource*, s'ha considerat interessant treballar-hi per estudiar mínimament l'estat de la recent implementació del llenguatge SPARQL (per exemple, com es veurà el l'apartat 4.3, no suporta la ordenació dels resultats) i la possible aportació al desenvolupament d'aquest projecte.

El factor determinant per treballar amb un repositori a memòria principal ve justificat per la poca documentació existent (només hi ha publicat un esborrany del manual per l'usuari i la documentació del seu l'API). A diferència de la documentació de l'anterior versió (on s'especifica com instal·lar la versió 1.2.6 sota les versions 4 o 5 del servidor Apache/Tomcat[30]), no hi ha cap especificació de com instal·lar la versió 2.0 sota cap DBMS. S'ha estudiat la documentació de la versió 1.2.6 per determinar la possible compatibilitat entre Apache/Tomcat i l'API de la versió 2.0 però davant la incertesa de que siguin compatibles s'ha decidit només treballar amb el repositori de memòria i deixar per a un possible continuació d'aquest treball la implementació sobre un servidor.

Per realitzar la implementació d'aquest prototipus també s'utilitzarà un entorn de desenvolupament integrat (IDE) per fer més còmoda la programació en Java.

Per tant, per la realització del prototipus per a consultes SPARQL sobre dades RDF es necessitarà disposar de les següents eines:

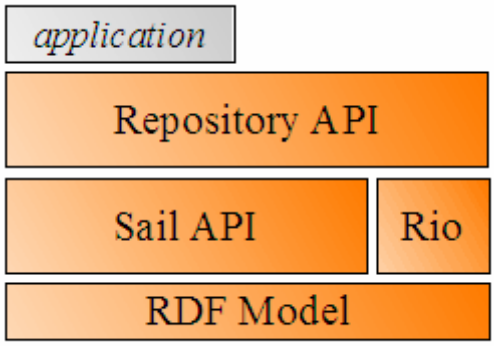
- API de Sesame 2.0 alpha 4 que suporta consultes amb el llenguatge SPARQL
- Instal·lació de Java 5 (especificada en els requeriments de la versió 2.0 de Sesame)
- Plataforma Eclipse[31], (un IDE de codi obert amb el que ja s'està familiaritzat per haver-la utilitzat amb d'altres assignatures) per realitzar la implementació.

[30] http://jakarta.apache.org/tomcat
[31] http://www.eclipse.org/

Per la utilització de Sesame només caldrà tenir una instal·lació de Java 5, copiar tots els fitxers del directori `/lib` de Sesame al director `/lib` de la instal.lació de Java i importar les seves llibreries dins de les classes de Java amb la comanda `import`.

5.2. DISSENY

Tal com s'ha indicat anteriorment es treballarà sobre el repositori a memòria que ofereix Sesame 2.0. L'arquitectura amb la que es treballarà serà la següent:



Donat que Sesame és un *framework* orientat a RDF la capa *RDF Model* proporciona defineix les interfícies i les instàncies per les entitats bàsiques de RDF.

La capa *RIO* ofereix en un conjunt de *parsers* i *writers* per traduïr els diferents formats de fitxers de dades RDF que existeixen. En aquest cas només s'utilitzarà la conversió de fitxer RDF/XML per carregar les dades al repositori de memòria.

La capa *SAIL API* (sistema API de baix nivell per l'emmagatzematge i l'anàlisi sintàctic de RDF) permet abstraure i fer inferència a partir de les dades guardades al repositori.

La capa *RESPOSITORY API* (API d'alt nivell que ofereix els mètodes per carregar fitxers de dades, fer consultes i extreure i manipular dades RDF) serà la capa que a nivell de disseny s'utilitzarà per poder desenvolupar una aplicació que treballi amb les dades RDF.

Com que les dades RDF s'emmagatzemaran a memòria, cada cop que es s'executi l'aplicació caldrà carregar les dades al repositori. Dels fitxers amb dades RDF que s'han proporcionat al taulell de l'assignatura se n'han triat els 10 següents a l'atzar:

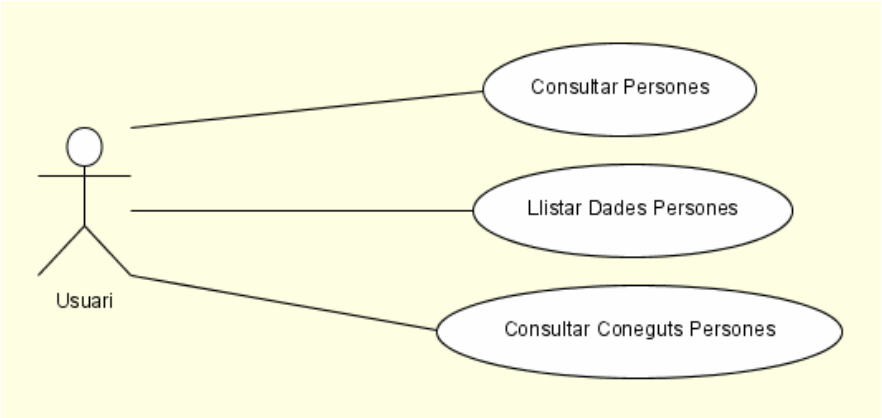
```
Alhabor.rdf, Bartek.rdf, csdk.rdf, Donald_Caine.rdf , egyr.rdf ,
foaf-main.rdf , foaf.rdf , Gral.rdf , Helga.rdf , Horacio.rdf.
```

Quan a l'aplicació es seleccioni l'opció de carregar dades es carregaran aquests fitxers al repositori de memòria.

Només es considerarà un únic actor que serà l'usuari de l'aplicació. Les consultes sobre dades RDF que podrà realitzar aquest actor seran les següents:

1. Consultar el nom de totes les persones
2. Per cada persona llistar el seu nom, *nick* i pàgina web (*homepage*).
3. Per cada persona llistar els noms dels seus coneguts

El diagrama de casos d'ús es el següent:



A continuació es fa l'especificació textual d'aquests tres cas d'ús:

Cas d'ús:	**Consultar Persones**
Resum de funcionalitats:	es mostren els noms de totes les persones
Actors:	Usuari
Precondició:	-
Postcondició:	No es mostren persones repetides.
Casos d'ús relacionats	-
Passos:	1. L'usuari demana consultar el nom de totes les persones. 2. El sistema mostra el nom de totes les persones.

Cas d'ús:	**Llistar dades Persona**
Resum de funcionalitats:	Es mostra el nom, nick i la pàgina web de totes les persones
Actors:	Usuari
Precondició:	-
Postcondició:	No es mostren registres de [nom, nick, pàgina web] repetits.
Casos d'ús relacionats	-
Passos:	1. L'usuari demana llistar el nom de totes les persones. 2. El sistema mostra el nom, nick i pàgina web de totes les persones.

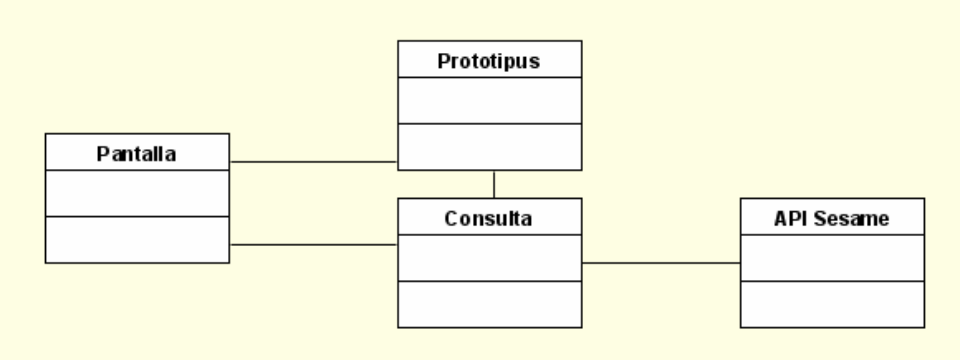
Cas d'ús:	**Consultar coneguts**
Resum de funcionalitats:	Per cada persona es mostra el nom de tots els seus coneguts.
Actors:	Usuari
Precondició:	-
Postcondició:	No es mostren registres de [nom, conegut] repetits.
Casos d'ús relacionats	-
Passos:	1. L'usuari demana consultar els coneguts. 2. Per cada persona en concret el sistema mostra el nom de la persona i el nom de tots els seus coneguts.

L'aplicació consistirà en un menú d'opcions que permetrà fer les següents seleccions:

- **Carregar dades**: l'aplicació carregarà al repositori de memòria els 10 fitxers RDF que s'han detallat anteriorment.
- **Conultar persones**: l'aplicació mostrarà per pantalla la llista de noms de totes les persones.
- **Consultar dades persones**: l'aplicació mostrarà per pantalla la llista de nom, *nick* i pàgina web de totes les persones.
- **Consultar coneguts persones**: l'aplicació mostrarà per pantalla la llista de nom de cada persona i el nom de tots els seus coneguts.

S'ha decidit que el programa tindrà les classes **Pantalla**, que s'encarregarà de la interacció de l'usuari (lectura del dispositiu d'entrada que en aquest cas serà el teclat, i escriptura del dispositiu de sortida que en aquest cas serà la pantalla), **Consulta**, que serà la classe encarregada de gestionar el respositori de memòria per carregar les dades i realitzar les consultes SPARQL mitjançant l'API de Sesame, i la classe **Prototipus** que gestionarà que la interacció amb l'usuari i l'execució de les consultes que demani en cada moment.

El diagrama de classes és el següent:



5.3. IMPLEMENTACIÓ

En aquest apartat primer es descriurà com es programa en Java la creació i gestió del repositori a memòria i com es fan les consultes SPARQL a través de l'API de Sesame 2.0. Després es veurà com funciona aquesta aplicació i com es mostren les consultes a l'usuari.

Descripció de l'API de Sesame 2.0

El primer pas que cal fer per a crear un repositori amb l'API de Sesame 2.0 és crear un objecte de tipus `Repository`. Aquest objecte es crea mitjançant el paquet `RepositoryImpl` que com el seu nom indica és la implementació del repositori. A Sesame es poden crear diferents tipus de repositori (a memòria, a memòria amb persistència a un fitxer...) i la creació del tipus de repositori depèn del paràmetre (objecte de tipus SAIL) que es passi a `RepositoryImpl` durant la creació de l'objecte `Repository`. Un cop creat el repositori cal inicialitzar-lo mitjançant el mètode `initialise()`per tal de crear l'estructura interna del repositori.

La creació d'un repositori a memòria implica que aquest repositori es volàtil, es a dir, que es perdrà el contingut el repositori un cop se surti del programa o que l'objecte `Repository` sigui destruït (quan deixi de ser referenciat) pel procés *garbage collection* de Java.

La creació d'un repositori per emmagatzemar dades RDF a memòria es fa de la següent manera:

```
import org.openrdf.sail.memory.MemoryStore;
import org.openrdf.repository.Repository;
import org.openrdf.repository.RepositoryImpl;

...

Repository myRepository = new RepositoryImpl(new MemoryStore());
myRepository.initialize();
```

Un cop creat el repositori a per poder-hi treballar cal crear un objecte de tipus `Connection` per establir una connexió amb el repositori a través del mètode `getConnection()`. L'objecte `Connection` representa una connexió amb el repositori i és l'objecte que proporciona els diferents mètodes per realitzar les diferents operacions amb el repositori (afegir dades i realitzar consultes principalment). Un cop acabades aquestes operacions amb el repositori cal tancar la connexió mitjançant la crida `close()`.

Per afegir dades al repositori existeixen diferents mètodes segons el format dels fitxers que contenen les dades RDF. El mètode que farem servir per el prototipus és el add() al que se li passen com a paràmetres l'objecte de tipus `File` que conté la referència al fitxer de dades RDF, la `baseURI` i el format del fitxer de dades RDF (en el nostre cas se li passarà el paràmetre `RDFFormat.RDFXML`).

En el següent exemple es mostra com afegir al les dades RDF contingudes a un fitxer RDF/XML:

```
import java.io.File;
import org.openrdf.repository.Connection;
import org.openrdf.repository.Repository;
import org.openrdf.repository.RepositoryImpl;
import org.openrdf.rio.RDFFormat;
import org.openrdf.sail.memory.MemoryStore;

...

File f = new File("C:/DadesFOAF/Alhabor.rdf");
String baseURI = "http://xmlns.com/foaf/TFC";

try {
     Connection con = myRepository.getConnection();
     con.add(f,baseURI,RDFFormat.RDFXML);
     con.close();
}
catch (IOException e2) { //tractament de l'error }
catch (UnsupportedRDFormatException e2) { //tractament de l'error }
catch (RDFParseException e2) { //tractament de l'error }
```

Per a realitzar consultes Sesame 2.0 disposa de diferents mètodes per avaluar-les. Existeixen dos tipus de consultes segons el tipus de resultat que produeixen: les consultes de tuples (que retornen com a resultat un conjunt de tuples on cada tupla representa un resultat) i les consultes de grafs (que permeten extreure subgrafs de les dades emmagatzemades al repositori). En aquest prototipus només es necessita realitzar consultes de tuples del primer tipus per poder mostrar els resultats de les consultes per pantalla en forma tabular.

L'objecte `Connection` disposa del mètode `evaluateTupleQuery` per avaluar una consulta al que se li passen com a paràmetres el llenguatge de consultes (en el cas de consultes SPARQL s'escriu `QueryLanguage.SPARQL`) i un `String` amb la consulta. Aquest mètode retorna la consulta en un objecte de tipus `TupleQueryResult` (resultats d'una consulta de tipus tupla). Aquest objecte conté una seqüència d'objectes de tipus `Solution` que contenen cadascuna de les solucions.

Per una altra part, l'objecte `Solution` conté el conjunt d'objectes `Binding` que guarda cada solució del resultat de la consulta. Per extraure cada solució, l'objecte `Solution` disposa del mètode `getValue("x")` on que retorna un objecte de tipus `Value` que conté la solució i que podem passar a String mitjançant la crida a `toString()`.

Un objecte `Binding` està format per un parell de noms relatius que contenen per a cada solució el nom de la solució (la variable de la consulta a la que dona solució) i la propia solució identificada per la variable "x" de la consulta.

L'objecte `TupleQueryResult` disposa del mètode `nextSolution()` per recorrer el conjunt de solucions de l'objecte i del mètode `isEmpty()` per saber si s'ha arribat al final.

En els següent exemple es mostra com es realitza una consulta SPARQL i com es fa el recorregut de cada resultat per extreure cadascuna de les solucions:

```
import java.io.File;
import org.openrdf.repository.Connection;
import org.openrdf.queryresult.TupleQueryResult;
import org.openrdf.queryresult.Solution;
import org.openrdf.model.Value;
import org.openrdf.querymodel.QueryLanguage;
import org.openrdf.queryresult.TupleQueryResult;
import org.openrdf.rio.RDFFormat;
...

Connection con = myRepository.getConnection();
    try {
    TupleQueryResult res =
        con.evaluateTupleQuery(QueryLanguage.SPARQL, "SELECT ?x ?y WHERE ?x p ?y");

    String s = res.getBindingNames().toString();
    // obté el nom de les variables de les que es mostren a la solució

    //iterar per cadascuna de les solucions
    while (!(res.isEmpty())){
         solution = res.nextSolution();
        Value v1 = solution.getValue("x");
        Value v2 = solution.getValue("x");
        // tractament de les solucions
    }
    con.close();
    } catch (SailException e) { //tractament de l'error }
}
```

Funcinonament del prototipus

A l'iniciar a l'aplicació es mostra un menú de les opcions que permet fer el programa.

Quan es selecciona la opció 1 el programa carrega els 10 fitxers FOAF al repositori de memòria i informa si la càrrega s'ha finalitzat correctament:

```
    O P C I O N S :
-------------------------------------------------
1.- Carregar dades RDF
2.- Consultar persones (noms)
3.- Consultar dades persones (nom, nick i pàgina web)
4.- Consultar coneguts d'una pesona (noms)

Escriu una opció (0 per sortir): 1
OPCIÓ 1: Carregant dades...
Dades carregades
Pulsa enter per continuar...
```

Quna l'usuari selecciona aquesta opció l'aplicació llegeix els fitxers de dades del directori `C:\DadesFOAF\` (aquest directori amb els fitxers es proporciona dins el fitxer `ajordana_producte.zip` del lliurament).

Si es selecciona l'opció 2 el programa mostra el nom de totes les persones que hi ha al repositori:

```
    O P C I O N S :
------------------------------------------------
1.- Carregar dades RDF
2.- Consultar persones (noms)
3.- Consultar dades persones (nom, nick i pàgina web)
4.- Consultar coneguts d'una pesona (noms)

Escriu una opció (0 per sortir): 2
OPCIÓ 2: Consultant nom de les persones...

        [name]
1       Dave Beckett
2       Edd Dumbill
3       Tim Berners-Lee
4       Dan Brickley
5       Libby Miller
6       Matt Biddulph
7       Bijan Parsia
8       Jim Hendler
9       Sean B. Palmer
10      Norm Walsh
11      Morten Frederiksen
12      Jo Walsh
13      Damian Steer
14      Dan Connolly
15      Jan Grant
16      Eric Miller
17      Phil McCarthy

S'han trobat 17 resultats
Pulsa enter per continuar...
```

En totes les consultes cada línia de resultat es numera amb un comptador i al final de la consulta s'informa del nombre total de resultats trobats. En el cas que hi hagués cap problema durant l'execució de qualsevol de les des consultes el programa informaria de l'error i seguiria el flux d'execució si li és possible.

Seleccionant l'opció 3 el programa mostra en nom, *nick* i la pàgina web dels resultats:

```
    O P C I O N S :
------------------------------------------------
1.- Carregar dades RDF
2.- Consultar persones (noms)
3.- Consultar dades persones (nom, nick i pàgina web)
4.- Consultar coneguts d'una pesona (noms)

Escriu una opció (0 per sortir): 3
OPCIÓ 3: Consultant nom, nick i pàgina web de les persones...

        [name, nick, homepage]
1       Jan Grant , jang , http://ioctl.org/jan/
2       Eric Miller , em , http://purl.org/net/eric/
3       Eric Miller , emiller , http://purl.org/net/eric/

S'han trobat 3 resultats
Pulsa enter per continuar...
```

L'opció 4 mostra per cada persona els noms de tots els seus coneguts:

```
    O P C I O N S :
------------------------------------------------
1.- Carregar dades RDF
2.- Consultar persones (noms)
3.- Consultar dades persones (nom, nick i pàgina web)
4.- Consultar coneguts d'una pesona (noms)

Escriu una opció (0 per sortir): 4
OPCIÓ 4: Consultant nom dels coneguts de les persones...

        [Nom, Conegut]
1       Dave Beckett , Edd Dumbill
2       Dave Beckett , Tim Berners-Lee
3       Dave Beckett , Dan Brickley
4       Dave Beckett , Libby Miller
5       Dave Beckett , Matt Biddulph
6       Dave Beckett , Bijan Parsia
7       Dave Beckett , Jim Hendler
8       Dave Beckett , Sean B. Palmer
9       Dave Beckett , Norm Walsh
10      Dave Beckett , Morten Frederiksen
11      Dave Beckett , Jo Walsh
12      Dave Beckett , Damian Steer
13      Dave Beckett , Dan Connolly
14      Dave Beckett , Jan Grant
15      Dave Beckett , Eric Miller
16      Dave Beckett , Phil McCarthy

S'han trobat 16 resultats
Pulsa enter per continuar...
```

Cal comentar que en les tres consultes s'eliminen dels resultats els repetits mitjançant la comanda `DISTINCT` del llenguatge SPARQL. No ha estat possible ordenar el resultats per nom ja que la versió 2.0 de Sesame no incorpora l'operativa per la comanda `ORDER BY` definida en l'estàndard del llenguatge SPARQL (al intentar executar les consultes apareix el missatge d'error `Query result ordering is not yet supported`). En la documentació de Sesame 2.0 s'indica que tampoc està disponible l'eliminació funcional del repositori.

NOTA: totes les pantalles dels exemples anteriors s'han basat amb el fitxer XML/RDF amb dades FOAF disponible a l'adreça http://www.dajobe.org/foaf.rdf també contingut en el directori `C:\DadesFOAF\` amb el nom de proves.rdf.

6. CONCLUSIONS

El projecte de la Web Semàntica, tutelat pel W3C, suposa una gran canvi en la manera de veure Internet. Si en l'actualitat entenem la web com una gran "biblioteca" el futur de la xarxa com a gran "base de dades" suposa un gran repte.

La implementació de la Web Semàntica, en la què el contingut de la web es definirà a sí mateix i per tant serà processat per les pròpies màquines, proporcionarà una gran canvi del que avui en dia entenem per web.

Actualment s'han resolt els primers passos d'aquest camí com són la definició de RDF per representar els recursos de la web, OWL com a llenguatge que permet crear ontologies per publicar i compartir coneixement a la xarxa, i SPARQL com a llenguatge de consultes de dades RDF.

Però es plantegen grans reptes que cal superar com són la manera d'emmagatzemar la informació continguda en la Web Semàntica. Les base de dades ja permeten emmagatzemar dades RDF malgrat la gran majoria ho faci en format RDF/XML i de manera relacional. Els projectes actuals es centren a desenvolupar bases de dades que permetin emmagatzemar dades directament en format RDF: s'estan desenvolupant les bases de dades RDF natives.

També cal desenvolupar i acordar ontologies que permetin la representació amb RDF de la informació de la xarxa, així com l'aparició de nombrosos serveis web sobre la Web Semàntica que permetin compartir i utilitzar dades de persones, empreses o grans comunitats a través de la xarxa.

En aquest sentit també cal destacar la recent adopció l'any passat de SPARQL com a proposta del W3C de llenguatge de consultes sobre dades RDF i en aquests moment estem assistint al desenvolupament dels sistemes de gestió de bases de dades d'aquest llenguatge de consultes, com ho demostra la recent versió de *Sesame 2.0* que és la primera versió d'aquest *framework* que seguin la recomanació del W3C suporta aquest llenguatge.

Cal destacar que Sesame permet la seva implementació en una gran varietat de repositoris, ja siguin a memòria amb o sense fitxer de persistència de dades, a bases de dades relacionals, emmagatzemant tripletes RDF o a nivell remot mitjançant diferents protocols, degut a la seva independència de qualsevol DBMS.

Un altre tema vital és com fer la transició de la web actual a la Web Semàntica. Per a que aquesta nova xarxa pugui desenvolupar-se serà necessari que d'entrada tingui compatibilitat amb la tecnologia actual. Han aparegut algunes teories de com *traduir* les instàncies de la web actual a instàncies de la Web Semàntica o bé generar dinàmicament pàgines en HTML a partir de les ontologies i les instàncies a la Web Semàntica, però encara queda molt per decidir.

Existeix un gran interès des de l'entorn empresarial, el sector públic i el món acadèmic per poder fer de la Web Semàntica una realitat ja que creuen que és una peça important per fer el progrés definitiu cap a la societat de la informació i el coneixement.

Les grans empreses i institucions com Oracle, HP, el MIT o la Comunitat Europea estan dedicant diners i esforços per treballar en el desenvolupament de la Web Semàntica i també esta apareixent programari lliure com MySQL o Sesame que també aporten la seva contribució.

Resumint, encara queda molta feina per fer: es necessita desenvolupar més tecnologia i desenvolupar aplicacions reals que posin en pràctica els principis de la Web Semàntica, que es publiquin més ontologies compartides i que facin que aquesta nova xarxa adquireixi la massa crítica imprescindible per fer-la realitat.

7. GLOSSARI

- **API (*Application Programming Interface*)**: conjunt d'especificacions de comunicació entre components software que consisteix en un conjunt de crides al sistema (funcions) que ofereixen accés als serveis del sistema des dels processos i que representa un mètode per aconseguir abstracció en la programació.

- **Bases de dades RDF natives (RDF:BD)**: bases de dades que emmagatzemen i gestionen la informació continguda en format RDF.

- **Bases de dades XML natives (XML:BD)**: bases de dades que emmagatzemen i gestionen la informació continguda en format XML.

- **BD Relacional**: base de dades que emmagatzema les dades segons el model relacional definint taules amb registres i columnes per representar la informació.

- **DBMS (*Data Base Management Sysyem*)**: software d'administració de base de dades utilitzat per gestionar una base de dades relacional.

- **DTD (*Document Type Definition*)**: document que especifica restriccions en l'estructura i la sintaxis d'una document XML per mantenir un format comú de dades de tots els documents XML que l'utilitzin.

- **Expressió RDF**: part d'una tripleta RDF. Pot ser un URI, un node buit o un literal.

- **FOAF (*Friend Of A Friend*)**: projecte dins de la Web Semàntica per descriure relacions entre persones mitjançant RDF per que puguin ser processades per màquines.

- ***Framework***: conjunt de classes que constitueix una aplicació incompleta i genèrica.

- **Graf patró**: conjunt de tripletes patró.

- **Graf RDF**: graf dirigit i etiquetat format per un conjunt de tripletes RDF en les que dos nodes corresponents al subjecte i l'objecte d'un predicat es connecten mitjançant un arc dirigit i etiquetat corresponent al predicat de la tripleta RDF.

- **HTML (*HyperText Markup Language*)**: llenguatge de marques d'hipertext. És el llenguatge estàndard de les pàgines web dissenyat per estructurar el format i presentació de text.

- **HTTP (*HyperText Transfer Protocol*)**: Protocol de transferència d'hipertext. És el sistema de transferència utilitzat per enviar i rebre dades a través de la web i/o aplicacions client/servidor.

- **IRI (*International Resource Identifiers*)**: generalització d'un URI que pot representarse entre <> o mitjançant el format prefix :etiqueta.

- **Jena**: framework Java per manipular models RDF desenvolupat per l'empresa *Hewlett-Packard Development Company*.

- **MySQL**: RDBMS de llicència lliure desenvolupat per l'empresa *MySQL AB*.
- **Node buit**: recurs que no conté cap IRI.
- **Objecte**: valor d'una propietat.
- **Ontologia**: especificació formal i explícita d'una conceptualització compartida i que defineix un domini.
- **Oracle**: RDBMS considerat com un dels sistemes de bases de dades més complerts fabricat per l'empresa *Oracle Corporation*.
- **OWL (*Web Ontology Language*)**: llenguatge de marques per publicar i compartir dades utilitzant ontologies a la WWW per facilitar un model de marcatge construït sobre dades RDF/XML.
- **PostgreeSQL**: RDBMS de llicència lliure i de codi obert.
- **Predicat**: relació o propietat definida per a un subjecte.
- **Propietat**: aspecte específic, característica, atribut o relació utilitzat per descriure un recurs.
- **OWL (*Web Ontology Language*)**: llenguatge per a la definició d'ontologies que estén a RDFS que és l'estàndard i que és utilitzat en la Web Semàntica per publicar i compartir dades a partir de les ontologies que defineix.
- **RDBMS (*Relational Data Base Management Sysyem*)**: DBMS d'una base de dades relacional.
- **RDF (*Resource Description Framework*)**: llenguatge desenvolupat pel W3C per a la representació de recursos de la web mitjançant tripletes subjecte–predicat–objecte per crear el nou model d'estructuració de la informació necessari per crear la Web Semàntica.
- **RDFS (*RDF Schema*)**: llenguatge que permet definir un domini especificant mecanismes per definir recursos, classes, jerarquies de classes i restriccions en la utilització de les classes i les relacions.
- **RDQL**: llenguatge de consultes sobre dades RDF propi de Jena.
- **Recurs**: qualsevol cosa que pot ser representada per una URI.
- **RQL**: llenguatge de consultes sobre dades RDF desenvulupat per *l'institut ICS-FORTH*.
- **Sentència**: combinació d'un recurs, una propietat i un valor (que pot ser un literal o un altre recurs) i que està formada per subjecte, predicat i objecte.
- **SeRQL**: llenguatge de consultes sobre dades RDF propi de Sesame.

- **Sesame**: framework de Java de codi obert per manipular models RDF desenvolupat per l'empresa *Aduna*.

- **SPARQL (*SPARQL Protocol and RDF Query Language*)**: especificació que el W3C ha proposat com a estàndard per a la Web Semàntica que consisteix en l'especificació d'un llenguatge de consultes (Query Language Specification), d'un format de resultats XML (Query Results XML Format) que descriu el format dels resultats de les consultes SPARQL i d'un protocol d'accés a dades remotes (Data Access Protocol) que descriu el protocol per fer consultes remotes sobre dades RDF.

- **SQL Server**: RDBMS relacional basat en el llenguatge SQL desenvolupat per l'empresa *Microsoft Corporation*.

- **Subjecte**: recurs que es vol descriure.

- **Tripleta**: element bàsic de la Web Semàntica que s'utilitza per denotar les classes, les propietats de les classes i els seus valors formada per un subjecte, un predicat i un objecte.

- **Tripleta patró**: tripleta que pot contenir variables de consulta com a subjecte, predicat i/o objecte separats per un espai i amb un punt al final.

- **URI (*Uniform Resource Identifier*)**: cadena de caràcters que permet identificar un recurs independentment del context.

- **URL (*Uniform Resource Locator*)**: subconjunt de les URI que a banda d'identificar un recurs proporciona els mitjans per localitzar-lo a través d'Internet.

- **Variable de consulta**: identificador per combinar amb expressions RDF.

- **W3C (*World Wide Web Consortium*)**: organisme presidit per Sir Tim Berners-Lee encarregat de produir estàndards per la WWW.

- **Web Semàntica**: extensió de la web actual ideada per Sir Tim Berners-Lee (inventor de la WWW) que introdueix descripcions explícites del significat, estructures i serveis que permeten que la informació sigui processable per les màquines.

- **WWW (*Wold Wide Web*)**: sistema d'hipertext que funciona sobre la xarxa Internet per l'enllaç d'informació a escala global. Consisteix en una pila de protocols entre els que es troben l'HTTP i l'HTML.

- **XML (*eXtensible Markup Language*)**: metallenguatge extensible d'etiquetes desenvolupat pel W3C que defineix el format estàndard per a l'estructuració de dades i d'informació.

8. BIBLIOGRAFIA I DOCUMENTACIÓ

Grigoris Antoniou, Frank van Harmelen; *A Semantic Web Primer*, Ed. The MIT Press.

Jaume Sistac, Rafael Camps, Dolors Costal, Elena Rodríguez González, Carme Martín, *Bases de dades I*, Universitat Oberta de Catalunya - XP05/05002/00492 Tercera edició: setembre 2005.

Jaume Sistac, Rafael Camps Paré, Pablo Costa, Dolors Costal ,Josep Maria Marco , Elena Rodríguez, Ramon Segre, Toni Urpí Tubella, *Bases de dades II*, Universitat Oberta de Catalunya - XP03/05053/02047 Segona edició: febrer 2004

Alfonso Egio ,*Introducción a la Web Semántica: Resource Description Framework*, http://es.geocities.com/alfonso_egio/rdf.pdf

Sean B. Palmer (Ramon Antonio Parada adaptado y traducido), *Introducción a la Web Semantica* – http://ramonantonio.net/contents/web_semantica

Steve Harris, *SPARQL query processing with conventional relational database systems*, http://www.openrdf.org/doc/papers/Sesame-ISWC2002.pdf

Dave Beckett, *SPARQL RDF Query Language Reference Card*, http://www.ilrt.bris.ac.uk/people/cmdjb/2005/04-sparql/

UMBC Semantic Web Reference Card - v2, http://ebiquity.umbc.edu/get/a/resource/94.pdf

Jeen Broekstra, Arjohn Kampman, Frank van Harmelen, *Sesame: A Generic Architecture for Storing and Querying*, http://www.openrdf.org/doc/papers/Sesame-ISWC2002.pdf

Deusdit A. Correa Cornejo, *Bases de Datos nativas en XML*, http://www.informatizate.net/articulos/bases_de_datos_nativas_en_xml_20020712.html

Ronald Bourret , *Consulting, writing, and research in XML and databases*, http://www.rpbourret.com/xml/XMLURLs.htm

Li Ding, Lina Zhouy, Tim Finin, Anupam Joshi, *How the Semantic Web is Being Used: An Analysis of FOAF Documents*, http://ebiquity.umbc.edu/_file_directory_/papers/120.pdf

Jorge Pérez , Marcelo Arenas, Claudio Gutierrez, *Semantics of SPARQL*, http://ing.utalca.cl/~jperez/papers/sparql_semantics.pdf

Brian McBride, Jena: Implementing the RDF Model and Syntax Specification, http://www.hpl.hp.com/personal/bwm/papers/20001221-paper/

Jeen Broekstra, Arjohn Kampman, Frank van Harmelen, *Sesame: A Generic Architecture for Storing and Querying*, http://www.openrdf.org/doc/papers/Sesame-ISWC2002.pdf

María Jesús Lamarca, *Hipertexto, el nuevo concepto de documento en la cultura de la imagen*, http://www.hipertexto.info/documentos/web_semantica.htm

Semantic Web
http://www.w3.org/2001/sw

Guía Breve de Web Semántica
http://www.w3c.es/Divulgacion/Guiasbreves/WebSemantica

Blog { ?sujeto ?predicado ?objeto } => "Web Semántica"@es,
http://f14web.com.ar/inkel/blog/

An introduction to RDF
http://www-128.ibm.com/developerworks/library/w-rdf/

Resource Description Framework (RDF)
http://www.w3.org/RDF

An introduction to RDF: Exploring the standard for Web-based metadata,
http://www-128.ibm.com/developerworks/library/w-rdf/

W3 Consortium
http://www.w3.org

OWL Web Ontology Language
http://www.w3.org/TR/owl-features/

SPARQL Query Language for RDF
http://www.w3.org/TR/rdf-sparql-query/

SPARQL Tutorial
http://jena.sourceforge.net/ARQ/Tutorial/index.html

SPARQLer
http://sparql.org/

Jena – A Semantic Web Framework for Java
http://jena.sourceforge.net/

Introduction to Jena
http://www-128.ibm.com/developerworks/java/library/j-jena/

Sesame
http://www.openrdf.org/documentation.jsp

RDF Gateway
http://www.intellidimension.com/default.rsp?topic=/pages/site/products/rdfgateway.rsp